VŠB – Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra informatiky

Maskování SQL Server databáze se zaměřením na T-SQL kód

Masking of an SQL Server Database with Focus on a T-SQL

2017

Bc. Filip Sikora

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Filip Sikora**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Maskování SQL Server databáze se zaměřením na T-SQL kód
Masking of an SQL Server Database with Focus on a T-SQL

Jazyk vypracování: čeština

Zásady pro vypracování:

Maskování databáze umožňuje skrýt privátní hodnoty v datech. Maskovanou databázi pak můžeme poskytnout třetím stranám, bez obavy z úniku dat. Pokud měníme původní data přímo v databázi, pak se nutně musí změnit také související kód v některých procedurách, funkcích, triggerech a pohledech, pokud chceme, aby tento kód fungoval jako dříve. Na Katedře informatiky vznikl prototyp nástroje umožnující jednoduché maskování databáze a souvisejících SQL příkazů. Cílem této práce je rozšířit tento nástroj o analýzu a maskování T-SQL kódů, který obsahuje proměnné, podmínky, cykly a kurzory.

Práce bude probíhat v následujících krocích:
1. Seznámení se stávající implementací maskovacího nástroje.
2. Seznámení se s možnostmi analýzy procedurálního T-SQL kódu.
3. Analýza, návrh a implementace modulu, který bude mít následující funkce:
a) analýza uloženého procedurálního kódu,
b) maskování uloženého procedurálního kódu s ohledem na aktuální maskování databáze,
c) validace úspěšnosti maskování procedurálního kódu pro konkrétní vstupní hodnoty.
4. Provedení testů nad vzorovou databází.

Seznam doporučené odborné literatury:

[1] Fung, Benjamin, et al. "Privacy-preserving data publishing: A survey of recent developments." ACM Computing Surveys (CSUR) 42.4 (2010): 14.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Radim Bača, Ph.D.**

Datum zadání: 01.09.2016

Datum odevzdání: 28.04.2017

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 28. 4. 2017 .....

Bc. Filip Sikora

ABSTRAKT

Tato diplomová práce pojednává o možnostech a způsobech maskování dat v relačních databázích. Následně také obsahuje popis implementace vlastního nástroje pro maskování dat přímo na úrovni relační databáze, konkrétně Microsoft SQL Server. Data v relační databázi ovšem nemůžeme pouze libovolně zamaskovat, je potřeba co nejpřesněji zachovat vzájemné vztahy mezi daty jednotlivých databázových tabulek. Takto maskovanou relační databázi poté můžeme poskytnout třetím stranám beze strachu o únik citlivých dat, např. bankovní transakce, evidence skladových zásob, záznamy o výplatách zaměstatnců a jiné důvěrné informace. Na trhu zatím neexistuje nástroj, který by takové maskování umožnil, proto započal vývoj tohoto nástroje.

KLÍČOVÁ SLOVA

SQL, T-SQL, Microsoft SQL Server, Data, Maskování, Maskování dat

ABSTRACT

This diploma thesis documents options and ways of masking data in relational databases. It also contains implementation description of own tool for masking data directly on relational database level, specifically Microsoft SQL Server. However data in relational database cannot be just arbitrarily masked, it is necessary to keep mutual relations between data contained in database tables as close as possible to original data. Such masked database can be thereafter provided to third parties without worrying about sensitive data compromisation, for example bank transactions, warehouse evidence, employers pay checks and other confidential informations. There is no such tool on the market, which would allow data masking directly on relational database level, that's why development of this tool begun.

KEY WORDS

SQL, T-SQL, Microsoft SQL Server, Data, Masking, Data masking

ZKRATKY A SYMBOLY

Zkratka	Význam	Český význam
API	Application Programming Interface	Rozhraní pro programování aplikací
CSV	Comma Separated Values	Hodnoty oddělené čárkou
FK	Foreign Key	Cizí klíč
PK	Primary Key	Primární klíč
SQL	Structured Query Language	Strukturovaný dotazovací jazyk
XML	eXtensive Markup Language	Rozšiřitelný značkovací jazyk

OBSAH

1 Úvod ... 1

1.1 Motivace ... 1

2 Maskování teoreticky ... 2

2.1 Typy maskování ... 2

2.1.1 Dynamické maskování ... 2

2.1.2 Statické maskování ... 3

2.2 Existující nástroje ... 5

2.2.1 SQL Server Dynamic Data Masking ... 6

2.2.2 CX-Mask ... 7

3 Vlastní nástroj pro maskování ... 8

3.1 Porovnání s CX-Mask ... 8

3.2 Architektura ... 9

3.2.1 Modul maskování databáze ... 9

3.2.2 Modul analýzy a maskování dotazu ... 13

3.2.3 Uživatelské rozhraní ... 18

4 Testování ... 23

4.1 Testovací dotazy ... 23

4.1.1 Dotaz 1 ... 24

4.1.2 Dotaz 2 ... 24

4.1.3 Dotaz 3 ... 25

4.1.4 Dotaz 4 ... 25

4.1.5 Dotaz 5 ... 25

4.1.6 Dotaz 6 ... 26

4.1.7 Dotaz 7 ... 26

4.1.8 Dotaz 8 ... 27

4.1.9 Dotaz 9 ... 27

4.1.10 Dotaz 10 ... 28

4.1.11 Dotaz 11 ... 28

4.1.12 Dotaz 12 ... 29

4.1.13 Dotaz 13 ... 29

4.1.14 Dotaz 14 ... 29

4.1.15 Dotaz 15 ... 30

4.1.16 Dotaz 16 ... 30
4.1.17 Dotaz 17 ... 31
4.1.18 Dotaz 18 ... 31
4.1.19 Dotaz 19 ... 31
4.1.20 Dotaz 20 ... 32
4.1.21 Dotaz 21 ... 32
4.1.22 Dotaz 22 ... 33
4.2 Zhodnocení testování ... 33
5 Problémy, možná vylepšení a budoucí vývoj ... 35
5.1 Problémy ... 35
5.1.1 Nemaskování některých atributů ... 35
5.1.2 Maskování textu ... 35
5.1.3 Operátor like ... 36
5.2 Možná vylepšení ... 37
5.2.1 Ošetření rozsahových podmínek ... 37
5.2.2 Možnost volby maskovaných sloupců ... 37
5.2.3 Více maskovacích funkcí s možností volby ... 38
5.2.4 Kompatibilita s SQL Serverem 2016 ... 38
5.3 Další vývoj ... 38
5.3.1 Uložené procedury, funkce a pohledy ... 38
5.3.2 Vývoj API ... 39
6 Závěr ... 40

1 ÚVOD

1.1 MOTIVACE

Můžeme se ptát, k čemu je vlastně maskování dat, nejen relačních databází, dobré. Předpokládejme následující situaci:

Pracujete ve společnosti, která se zabývá průmyslovou výrobou. Mezi vašimi zákazníky jsou i zákazníci z automobilového průmyslu. Pro tyto zákazníky vyrábíte součástky na prototyp vozidla, které ještě nebylo uvedeno na trh, a zákazníkovi jste se zavázal mlčenlivostí o tomto projektu. Samozřejmě vedete skladovou evidenci, nicméně aktuální skladová evidence je, vzhledem k růstu vaší společnosti, silně nedostačující. Rozhodnete se proto, že si necháte implementovat nový systém skladové evidence. Společnost, která bude implementovat nový systém (třetí strana) by chtěla mít k dispozici váš aktuální systém, aby na jeho základě co nejlépe vyhověla vaším požadavkům. Vy ale třetí straně nemůžete předat systém v aktuálním nemaskovaném stavu, protože obsahuje i informace o projektu, vůči kterému jste vázáni mlčenlivostí. V této chvíli přichází na řadu maskování dat v aktuálním systému.

Maskování dat jako takové je velice jednoduché a přímočaré. V případě relačních databází ale musíme vzít v potaz i relace mezi daty v jednotilvých tabulkách. Pokud bychom všechna data jen libovolně zamaskovali, mohli bychom tím silně narušit vazby mezi daty v tabulkách. Naší snahou proto je, provést co nejlepší zamaskování dat, ale stále s ohledem na co nejvěrohodnější velikost výsledku vůči původní nemaskované relační databázi.

Co nejlepšího výsledku se snažíme dosáhnout nad vybranými testovácími dotazy, jelikož není v našich silách, aby byla celá databáze maskována a i přesto byly zachovány úplně všechny vztahy mezi daty bez narušení. S určitou úrovní narušení vztahů mezi daty je potřeba počítat vždy a je vhodné si předem určit, jaká úroveň narušení těchto vztahů je ještě přípustná.

2 MASKOVÁNÍ TEORETICKY

2.1 TYPY MASKOVÁNÍ

Samozřejmě existuje více druhů maskování dat. Podíváme se na ty nejpoužívanější, případně na ty, které používá už nějaký existující nástroj.

2.1.1 DYNAMICKÉ MASKOVÁNÍ

Dynamické maskování nemaskuje data na úrovni relační databáze, data v databázi zůstavájí beze změny. Maskování se použije až v případě dotazu vůči relační databázi[1]. Očividná výhoda tohoto přístupu je, že zůstávají plně zachovány vztahy mezi daty v tabulkách relační databáze. Další výhoda je, že můžeme, na úrovni relační databáze nastavit, kteří uživatelé uvidí data jako maskovaná a kteří uvidí data v původní podobě. Také je možné, za předpokladu, že třetí strana nebude do dat zasahovat, tzn., nebude provádět operace INSERT, UPDATE, DELETE, atp., (můžeme například odebrat právo WRITE) dát k dispozici přímo aktuální systém.

Nevýhodou tohoto přístupu je potřeba, aby dotazy nad relační databází obsahovaly v podmínkách správné (nemaskované) hodnoty. To proto, že data na úrovni relační databáze nejsou maskovaná.

Například dotaz včetně ukázkového výsledku:

```
SELECT * FROM CUSTOMER
```

C_ID	C_NAME	C_ADDRESS	C_CREDITCARD	C_SALARY
1	Jane Doe	Address 1	‚xxxx xxxx xxxx xxx‘	0
2	John Smith	Address 2	‚xxxx xxxx xxxx xxx‘	0
…	…	…	…	…

Atributy C_CREDITCARD a C_SALARY byly maskovány.

Vidíme, že je výsledek správně zamaskován, ale co když budeme potřebovat dotaz, ve kterém vybíráme zákazníka na základě kreditní karty? Třetí straně nemůžeme poskytnout seznam čísel kreditních karet pro dotazování.

Z tohoto problému přímo vyplývá další problém a tím je možnost hádání, například metodou brute-force, hodnoty maskovaných dat.

Upravme dotaz výše následovně:

```
SELECT * FROM CUSTOMER WHERE C_CREDITCARD = '1111 2222 3333 4444'
```

C_ID	C_NAME	C_ADDRESS	C_CREDITCARD	C_SALARY
2	John Smith	Address 2	‚xxxx xxxx xxxx xxx‘	0

Atributy C_CREDITCARD a C_SALARY byly maskovány.

Opět vidíme, že je výsledek správně maskován, ale je zřejmé, že nyní je již maskování naprosto zbytečné, jelikož uživatel zná číslo kreditní karty přímo z podmínky dotazu. Pokud by si třetí strana

vytvořila nástroj pro automatické zkoušení všech všech kombinací, tzv. brute-force nástroj, byla by schopna relativně rychle a jednoduše zjistit čísla kreditních karet všech zákazníků. V případě kreditních karet je to o to jednodušší, jelikož formát čísla kreditní karty je přesně daný a obsahuje pouze číslice. Jednou z možností, jak brute-force zamezit je, omezit na úrovni relační databáze počet dotazů za jednotku času. Tímto ale zase zamezíme třetí straně provést výkonnostní testování, které je pro úspěšné nasazení nového systému nezbytné.

Případně rozsahový dotaz:

```
SELECT * FROM CUSTOMER WHERE C_SALARY > 49500 AND C_SALARY < 50500
```

C_ID	C_NAME	C_ADDRESS	C_CREDITCARD	C_SALARY
1	Jane Doe	Address 1	‚xxxx xxxx xxxx xxx‘	0

Atributy C_CREDITCARD a C_SALARY byly maskovány.

Výsledek je opět správně maskován, ale z podmínky vyplývá, že plat zákazníka je v intervalu 49 500 až 50 500. Proti tomuto úniku dat se v podstatě nelze chránit vůbec, neboť i s malým počtem dotazů za jednotku času můžeme zjistit dostatek informací - nehledáme konkrétní hodnoty.

V návaznosti na výkonnostní testování je potřeba dodat, že i samo maskování dat má určitou výpočetní složitost, a jelikož se, v případě dynamického maskování, provádí pro každý dotaz, bude i samo maskování negativně ovlivňovat výkonnost systému. Samozřejmě ne tak drasticky, jako v případě omezení počtu dotazů za jednotku času, ale musíme počítat, že výkonnost systému bude nižší.

Dynamicky maskovaná data také nemůžeme fyzicky předat třetí straně, tím bychom ztratili jakoukoliv kontrolu nad daty. Můžeme pouze poskytnout rozhraní s výše uvedenými omezeními.

Výhody:

- Není potřeba kopie databáze, data se maskují až před vrácením výsledku dotazu
- Nenarušuje relace mezi daty v tabulkách
- Možnost volby, pro které uživatele budou data maskována a pro které ne
- Za určitých okolností může třetí strana přistupovat přímo k databázi systému

Nevýhody:

- Potřeba nastavit pro každého uživatele zvlášť
- Problém s hodnotami pro podmínky v dotazu
- Kompromitující dotazy a případně brute-force útok
- Ovlivňuje výkon každého dotazu
- Nemožnost fyzického předání databáze třetí straně

2.1.2 STATICKÉ MASKOVÁNÍ

Statické maskování, na rozdíl od dynamického maskování, se zakládá na fyzickém maskování dat přímo na úrovni relační databáze. Na začátek je potřeba vytvořit kopii relační databáze i s původními daty.

Dále zvolíme, která data budou maskovaná a to buď automaticky, nebo manuálně. Následně se provede maskování vybraných dat na kopii relační databáze[2]. Nyní se musí maskovat hodnoty ve vybraných testovacích dotazech tak, aby co nejvěrněji odpovídaly maskovaným hodnotám v maskované kopii relační databáze.

Pro příklad si vezmeme stejný dotaz jako v případě dynamického maskování, včetně ukázkového výsledku:

```
SELECT * FROM CUSTOMER
```

C_ID	C_NAME	C_ADDRESS	C_CREDITCARD	C_SALARY
1	Jane Doe	Address 1	‚aaaa bbbb cccc dddd‘	6481235
2	John Smith	Address 2	‚eeee ffff gggg hhhh‘	4271
…	…	…	…	…

Atributy C_CREDITCARD a C_SALARY byly maskovány.

Vidíme, že výsledek je přímo maskován, zde není žádný problém. Uvažujme tedy stejnou situaci, jako v případě dynamického maskování. Tzn., hledáme zákazníka na základě čísla kreditní karty.

Mějme původní dotaz s podmínkou:

```
SELECT * FROM CUSTOMER WHERE C_CREDITCARD = '1111 2222 3333 4444'
```

Pokud i tento testovací dotaz zamaskujeme nástrojem pro maskování, měli bychom dostat následující dotaz se správným výsledkem:

```
SELECT * FROM CUSTOMER WHERE C_CREDITCARD = 'eeee ffff gggg hhhh'
```

C_ID	C_NAME	C_ADDRESS	C_CREDITCARD	C_SALARY
2	John Smith	Address 2	‚eeee ffff gggg hhhh‘	4271

Atributy C_CREDITCARD a C_SALARY byly maskovány.

Takto maskovaná relační databáze je, pokud byla použita vhodná a bezpečná maskovací funkce, zabezpečená proti úniku citlivých dat, a to i metodou brute-force, a tudíž může být bez problému předána třetí straně.

Momentálně ovšem neexistuje nástroj, který by toto komplexní, tzn. včetně dotazů, uložených procedur, triggerů atp., jednoduše umožňoval. Uložené procedury a funkce mohou ve svém kódu obsahovat konstanty, které je také potřeba maskovat.

Výše byl velice jednoduchý dotaz pro ukázku, nicméně hned další, rozsahový dotaz s sebou přinese určité komplikace.

Připomeňme si tedy původní rozsahový dotaz:

```
SELECT * FROM CUSTOMER WHERE C_SALARY > 49500 AND C_SALARY < 50500
```

Pokud tento dotaz zamaskujeme nástrojem pro maskování, můžeme očekávat následující případy:

- `SELECT * FROM CUSTOMER WHERE C_SALARY > 564732 AND C_SALARY < 3791`
  Podmínka je vždy neplatná, protože rozsah není platný. Velikost výsledku je nula.

- `SELECT * FROM CUSTOMER WHERE C_SALARY > 3791 AND C_SALARY < 564732`
  Podmínka je platná pro příliš mnoho záznamů. Velikost výsledku je příliš velká.

- `SELECT * FROM CUSTOMER WHERE C_SALARY > 3971 AND C_SALARY < 4395`
  Ideální stav, rozsah podmínky přibližne odpovídá nemaskované podmínce. Velikost výsledku je přibližně stejná s původní velikostí výsledku. Ovšem toto nelze jednoduše a jednoznačně zaručit.

I na tomto, velice jednoduchém dotazu, byly demonstrovány problémy, které mohou při statickém maskování dat nastat. Tyto problémy se stupňují se složitostí dotazu a složitostí modelu relační databáze. A to ani zadleka nejsou ty největší problémy. Největším problémem v dotazích nad staticky maskovanou relační databází jsou jednoznačně funkce v dotazech a případně operátor „LIKE“. Problémem také může být, dle situace, že data jsou maskována vždy a pro všechny. Také každý nový testovací dotaz nebo jejich množina musí být zvlášť maskována před předáním třetí straně.

Samozřejmě má statické maskování i své výhody. Hlavní výhoda již byla zmíněna a tou je možnost bezpečně předat takto maskovanou databázi třetí straně beze strachu o únik citlivých dat. Bezpečnost takto maskovaných dat závisí na typu maskovací funkce pro jednotlivé datové typy. Statické maskování také nestojí žádný výpočetní výkon navíc, jelikož data jsou již maskována a není tudíž ovlivněn výkon relační databáze. Maskování se provede jen jednou a to pro celou relační databázi najednou. Taková databáze je tedy vhodná i pro výkonnostní testování.

Výhody:

- Bezpečnost, možnost předat třetí straně celou relační databázi
- Neovlivňuje výkon
- Bez problému s hodnotami podmínek v dotazech – nelze získat kompromitující informace ani rozsahovými dotazy

Nevýhody:

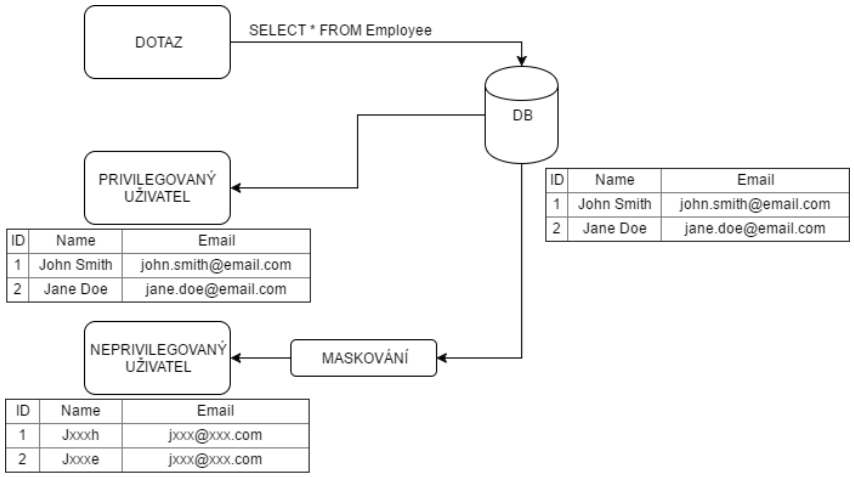
- Může, a v případě složitějších dotazů a modelů relační databáze skoro jistě, ovlivňuje velikost výsledku
- Vždy je potřeba kopie relační databáze
- Maskováno vždy a pro všechny
- Narušuje relace mezi daty v tabulkách
- Zatím neexistuje nástroj umožňující komplexní maskování – tzn. včetně dotazů, uložených procedur, pohledů, triggerů atp.

2.2 EXISTUJÍCÍ NÁSTROJE

Tato kapitola se věnuje možnostem, výhodám a nevýhodám exitujících nástrojů pro maskování relačních databází.

2.2.1 SQL SERVER DYNAMIC DATA MASKING

Nativní dynamické maskování dat dostupné pro Microsoft SQL Server (od verze 2016) a Azure SQL Database. Není potřeba žádného nástroje třetí strany, maskování je přímo součástí Microsoft SQL Serveru a Azure SQL Database. Jedná se o standardní dynamické maskování se všemi jeho výhodami i nevýhodami.[3]



Obrázek 1: SQL Server DDM (dynamické maskování)

Toto rozšíření obsahuje předdefinované funkce pro maskování nejčastějších druhů citlivých dat, jako například maskování emailové adresy.

Předdefinované funkce[3]:

- **Default** – text je celý maskován znaky ‚X‘, číslo je vždy maskováno jako 0, datum je vždy maskováno jako 1. 1. 1900 a binární data jsou maskována jedním byte s decimální hodnotou 0
- **Email** – pro maskování emailových adres, z původní emailové adresy je odmaskováno první písmeno, dále je odmaskován zavináč a nakonec doména nejvyššího řádu konstatní hodnoty ‚.com‘. Zbytek emailové adresy je maskován znaky ‚X‘ Například emailová adresa ‚abcd@efgh.net‘ je maskována jako ‚aXXX@XXX.com‘.
- **Random** – pouze pro číselné hodnoty, je nutno specifikovat možné minimum a maximum náhodných hodnot.
- **Custom String** – funkce, přebírající 3 parametry. Hodnota prvního parametru určuje, kolik znaků ze začátku původního textu nebude maskováno. Hodnota (text) druhého parametru je přidána k prvnímu parametru. Poslední parametr říká, kolik znaků od konce původního textu nebude maskováno. Například, pokud požijeme tuto funkci s parametry (1, ‚xxx‘, 1) nad hodnotou ‚HELLO WORLD‘ maskovaná hodnota bude ‚HxxxD‘.

Nastavení nástroje je, vzhledem k jeho nativitě, velice jednoduché:

- Definice sloupce v příkazu „CREATE TABLE“:
  Email varchar(100) MASKED WITH (FUNCTION = 'email()')
- Je také možno nastavit maskování pro již existující sloupec tabulky:
  ALTER COLUMN Email ADD MASKED WITH (FUNCTION = 'email()')
- Také můžeme zrušit maskování již maskovaného sloupce:

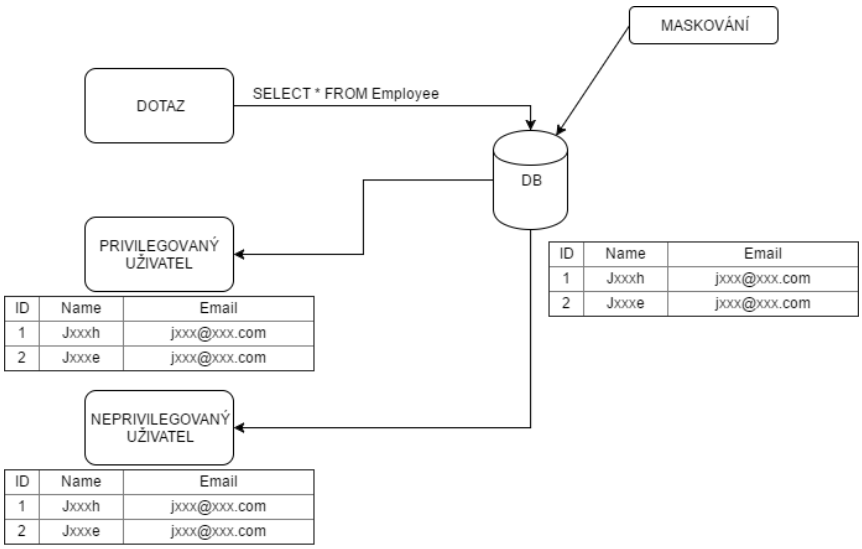
ALTER COLUMN Email DROP MASKED;

- Nastavení uživatelského účtu pro zobrazení dat bez maskování:
  GRANT UNMASK TO TestUser;
- Nastavení uživatelského účtu pro zobrazení maskovaných dat:
  REVOKE UNMASK TO TestUser;

2.2.2 CX-MASK

Multiplatformní nástroj od firmy Camouflage, podporuje většinu běžně používaných relačních databází. Pro účely maskování dat využívá statického maskování. Data jsou maskována pomací tzv. datasetů, to jsou soubory obsahující určitý druh dat, například jména, adresy, atd. Vybrané a nejčastěji používané datasety, jako jsou jména, adresy, atd., jsou sočástí aplikace[4]. CX-Mask se snaží o co nejrealističtější maskování dat tak, aby data po procesu maskování nevypadala uměle. Nástroj ovšem nepodporuje maskování parametrů testovacích dotazů, spoléhá na realističnost dat a využití přímo koncovou aplikací.

Nástroj se proto jeví vhodný například pro maskování dat do testovacího systému pro trénink nováčků – zaměstnanců. Zaměřuje se spíše na maskování vybraných sloupců v tabulkách relační databáze pseudo-reálnými daty, není úplně vhodný k maskování celé relační databáze jako takové.



Obrázek 2: CX-Mask (statické maskování)

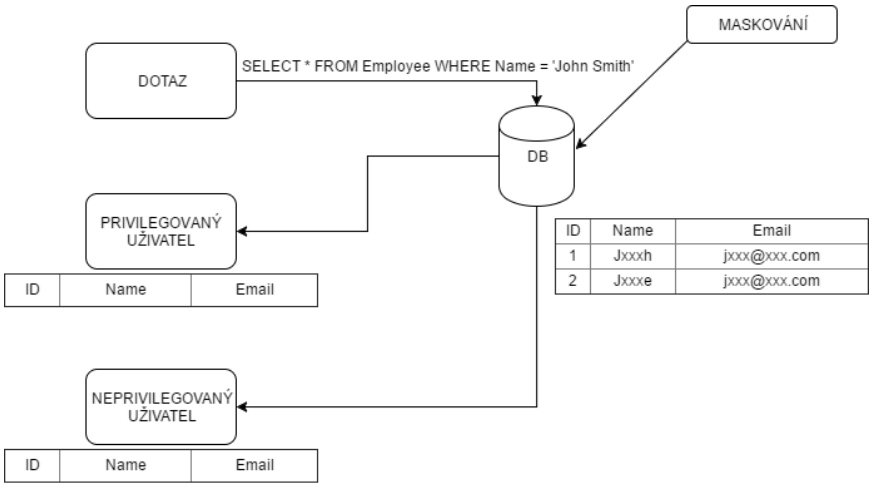
3 VLASTNÍ NÁSTROJ PRO MASKOVÁNÍ

Na katedře informatiky vznikly komponenty umoňující provedení elementárních kroků při anonymizaci databáze a jejího vytížení. Mým úkolem bylo spojit tyto komponenty do jedné aplikace, odladit případné chyby v komponentách a přidat ruční implementaci, která umožní jednoduše dokončit to, co se nepovede maskovat s pomocí existujících komponent. V kapitolách 3.2.1 Modul maskování databáze a 3.2.2 Modul analýzy a maskování dotazu si představíme komponenty, ze kterých jsem v řešení vycházel a v kapitole 3.2.3 Uživatelské rozhraní popíšeme výslednou aplikaci.

3.1 POROVNÁNÍ S CX-MASK

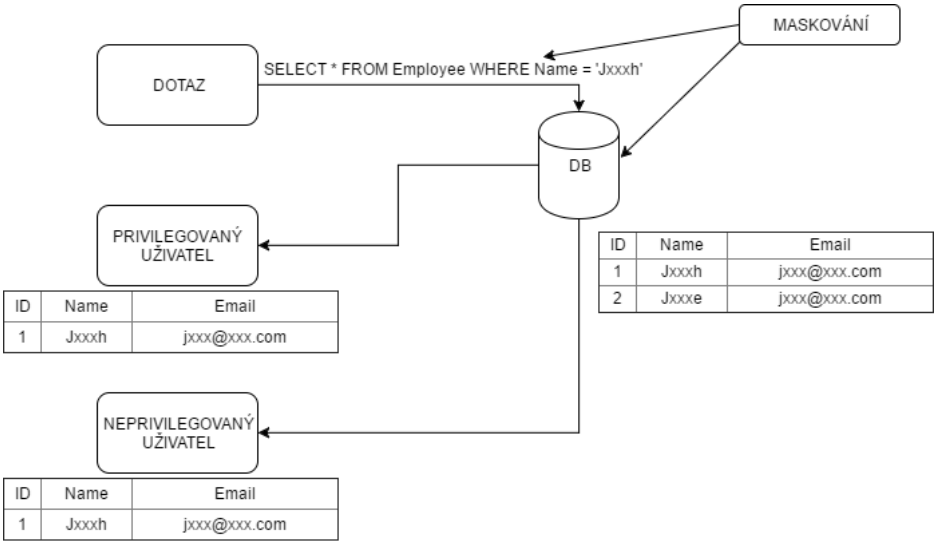
Na začátek je potřeba vysvětlit, proč vůbec vyvíjet nový a složitý nástroj, když už existuje CX-Mask, který statické maskování umí. Uvažujme ukázkový příklad z Obrázek 2: CX-Mask (statické maskování). Takové maskování je v pořádku a funguje správně. Také bude, s velkou pravděpodobností, fungovat správně v koncovém systému, kde se budou podmínky dotazů generovat dynamicky dle volby uživatele. Nicméně může nastat situace, kdy je potřeba maskovat i podmínky dotazu.

Na takovou situaci můžeme narazit při předávání relační databáze třetí straně a to v případě, že nechceme, nebo nemůžeme předat celý systém jako takový, například z právních, či technických důvodů. V takovém případě můžeme třetí straně předat relační databázi s množinou dotazů. Tato množina by měla obsahovat časté, časově náročné a další důležité dotazy, se kterými koncový systém pracuje.



Obrázek 3: Statické maskování (bez maskování dotazu)

Vidíme, že výsledek z Obrázku 3 je špatně a je nutné maskovat i hodnoty v podmínce dotazu. Zavedeme tedy maskování i na samotný dotaz, jak je zobrazeno na Obrázek 4: Statické maskování (včetně maskování dotazu).



Obrázek 4: Statické maskování (včetně maskování dotazu)

A nyní již správně, včetně maskování podmínky dotazu. Náš nástroj by měl přesně toto umožňovat, samozřejmě vzhledem k maskovaným hodnotám v relační databázi tak, aby byla velikost výsledku nemaskovaného dotazu vůči nemaskované relační databázi co nejblíže velikosti výsledku maskovaného dotazu vůči maskované relační databázi. Jak už bylo zmíněno výše, dosáhnout spolehlivě správného výsledku není vůbec jednoduché a ani naše aplikace, jakožto koncept, to zatím spolehlivě nezvládá.

3.2 ARCHITEKTURA

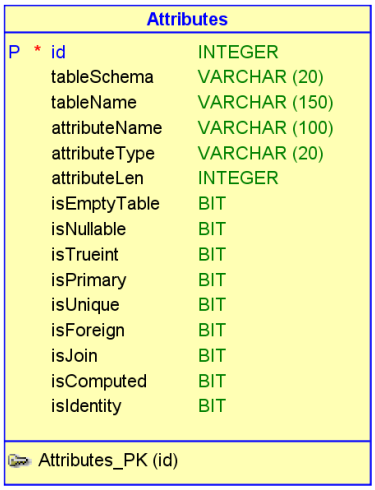
3.2.1 MODUL MASKOVÁNÍ DATABÁZE

Seznam funkcí

- 3.2.1.1 – Funkce pro vytvoření metadat
- 3.2.1.2 – Funkce pro maskování databáze
- 3.2.1.3 – Funkce maskování hodnoty
- 3.2.1.4 – Funkce pro načtení metadat

Tento modul zastřešuje funkce pro statické maskování vlastní databáze a obsahuje elementární funkce pro maskování jednotlivých datových typů. Tento modul také pracuje s tzv. Metadaty, to je separátní databáze, ve které jsou uloženy všechny názvy sloupců jednotlivých tabulek a k nim jsou uloženy základní informace, např. zda se jedná o primární či cizí klíč, zda může být hodnota NULL atp., viz Obrázek 5: Schéma tabulky s metadaty.

Na základě těchto metadat je posléze určeno, které atributy budou maskovány a které ne. V aktuální implementaci jsou z maskování vynechány pouze primární a cizí klíče, ostatní atributy jsou automaticky maskovány.



Obrázek 5: Schéma tabulky s metadaty

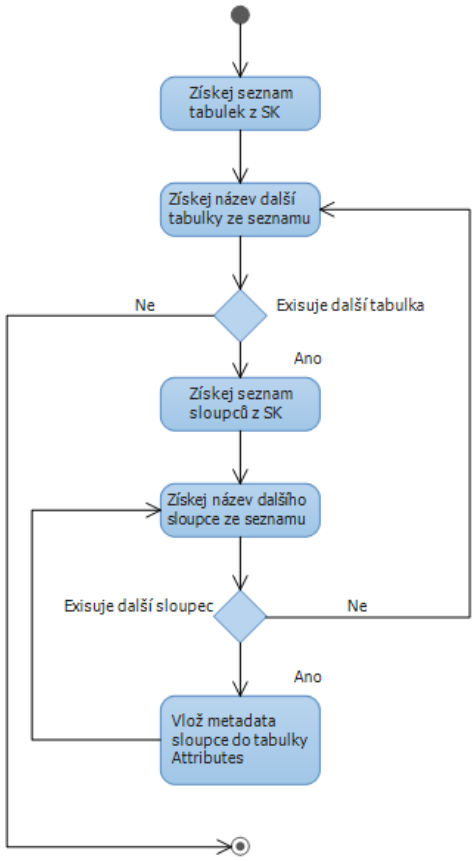
Metadata musí být vytvořena ještě před samotným maskováním databáze. Následně je, na základě metadat, maskována samotná relační databáze.

3.2.1.1 FUNKCE PRO VYTVOŘENÍ METADAT

Vstupy: Připojení k původní databázi, připojení k databázi metadat

Výstupy: Výstupem funkce jsou vytvořená metadata v databázi

Nejprve proběhne připojení ke zvolené databázi. Ze systémového katalogu je získán seznam všech tabulek v databázi. Pro každou tabulku je následně získán, ze systémového katalogu, seznam všech sloupců. Pro každý sloupec každé tabulky je poté vytvořen záznam v tabulce Attributes. Tato funkce již byla implementována a v rámci mé práce nebyla nijak modifikována.



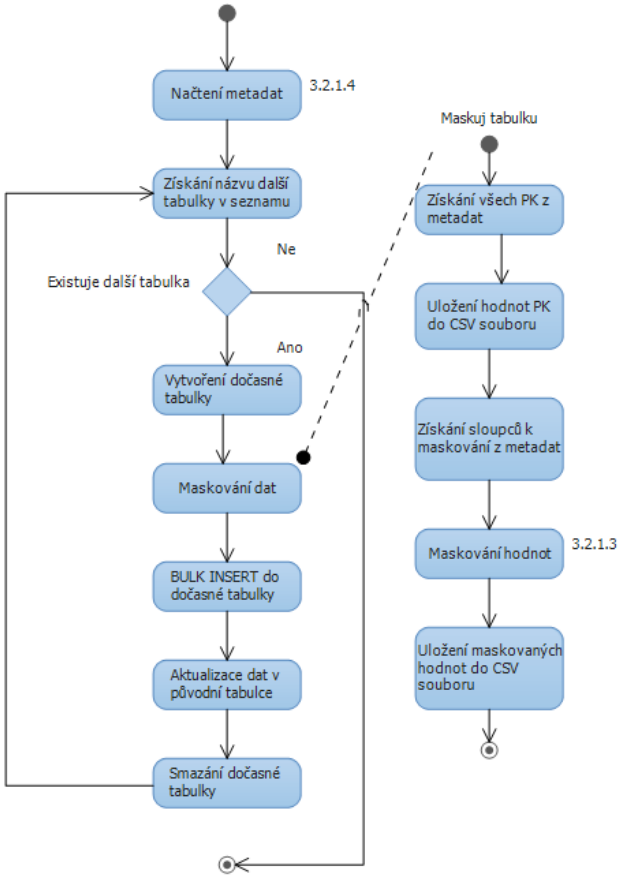
Obrázek 6: Aktivity diagram vytvoření metadat

3.2.1.2 FUNKCE PRO MASKOVÁNÍ DATABÁZE

Vstupy: Připojení k databázi s metadaty, připojení k databázi k maskování

Výstupy: Výstupem funkce je maskovaná databáze

Funkce pro statické maskování zvolené databáze. K chodu potřebuje dva parametry, připojení k metadatům a připojení k cílové databázi k maskování. Jako první jsou načtena metadata, která předtím musejí být vytvořena pomocí: 3.2.1.1 Funkce pro vytvoření metadat. Z metadat získáme názvy všech tabulek. Pro každou takto získanou tabulku pak zjistíme z metadat seznam primárních klíčů a seznam sloupců k maskování. Sloupce k maskování jsou vybírány na základě metadat a jde o sloupce, které nejsou PK, FK, Join, Computed, Identity, Unique a zároveň datový typ sloupce není binary, varbinary nebo neznámý datový typ. Získané tabulky jsou poté sekvenčně zpracovány.



Obrázek 7: Aktivity diagram maskování databáze

Zpracování probíhá tak, že je vytvořena dočasná tabulka tak, aby v ní mohla být uložena data aktuálně zpracovávané tabulky. Ukládájí se pouze primární klíče a hodnoty, u kterých bylo určeno, že budou maskovány. Data ze zpracovávané tabulky jsou přečtena a uložena do CSV souboru. Před uložením dat do CSV souboru jsou data, na základě metadat, maskována. Z tohoto souboru jsou poté, operací BULK INSERT, vložena do dočasné tabulky. Z toho plyne omezení, že aplikace musí být spuštěna na stejném počítači, jako je samotná databáze, jelikož Microsoft SQL Server neumožňuje, jako vstup operace BULK INSERT, jiný soubor, než lokální. Nakonec jsou data v zdrojové tabulce nahrazena daty z dočasné tabulky a po skončení této operace je dočasná tabulka smazána. Celý tento proces se opakuje pro všechny tabulky ze seznamu.

Tato funkce již byla implementována, ale v rámci mé práce byly opravovány nalezené chyby nebo rozšiřována funkčnost. Ve funkci bylo opraveno špatné maskování datového typu „char“ v maskované databázi a z toho vyplývající následná nesprávnost výsledku. Dále jsem opravoval menší chyby a upravil funkci tak, aby se dala použít z GUI.

3.2.1.3 FUNKCE MASKOVÁNÍ HODNOTY

Vstupy: nemaskovaná hodnota

Výstupy: maskovaná hodnota

Funkce pro maskování hodnot na vstupu přebírají původní hodnotu a vracejí maskovanou hodnotu. Funkce se liší dle datového typu, který chceme maskovat. Vhodná funkce je zvolena na základě datového typu hodnoty v tabulce, ten získáme z metadat.

V aktuální implementaci jsou funkce pro maskování velice prosté a definitivně nejsou bezpečné. Jedná se ovšem o koncept a na bezpečnosti nám momentálně nezáleží. Maskovací funkce se dají, v případě potřeby, jednoduše nahradit jinými, bezpěčnějšími. Tato funkce již byla implementována a v rámci mé práce nebylo momentálně cílem ji nijak modifikovat.

3.2.1.4 FUNKCE PRO NAČTENÍ METADAT

Vstupy: Připojení k databázi metadat

Výstupy: Metadata pro využití v aplikaci

Funkce se připojí k databázi metadat pomocí předaného připojení a z tabulky Attributes načte všechna metadata do paměti. Tato funkce již byla implementována a v rámci mé práce nebyla nijak modifikována.

3.2.2 MODUL ANALÝZY A MASKOVÁNÍ DOTAZU

Seznam funkcí:

- 3.2.2.1 - Funkce pro zpracování dotazů
- 3.2.2.2 - Funkce načtení dotazů ze souboru
- 3.2.2.3 - Funkce maskování dotazu
- 3.2.2.4 - Funkce analýzy dotazu
- 3.2.2.5 - Funkce porovnání velikosti výsledků a exekučních plánů

Tento modul zastřešuje operace kolem maskování testovacího vytížení (dotazů) vůči maskované relační databázi. V prvním kroku je přečten a rozparsován vstupní soubor s vytížením, jsou nalezeny jednotlivé dotazy. Následně je každý dotaz analyzován. Dotaz se jako první vykoná v původní (nemaskované) variantě. Pro dotaz si necháme relační databází vrátit jeho exekuční plán ve formátu XML. Tento plán je poté analyzován rekurzivním sestupem na výskyt hodnot k maskování. Pokud nalezneme hodnotu, z exekučního plánu zjistíme, zda je to konstanta, nebo hodnota náleží k nějakému atributu. Pokud patří k atributu, je název atributu uložen společně s nalezenou hodnotou. Poté, co jsou takto analyzovány všechny dotazy z vytížení, je pro každý dotaz uložena kolekce nalezených hodnot. Z této kolekce následně vybíráme jednotlivé hodnoty, a pokud hodnota náleží nějakému atributu, v metadatech se zkontroluje, zda má být tato hodnota maskována. Pokud ano, provedeme její maskování v závislosti na jejím datovém typu. Nakonec je dotaz opět sestaven z původního dotazu a případných nových maskovaných hodnot.

Tato funkce již byla implementována, ale v rámci mé práce byly opravovány nalezené chyby nebo rozšiřována funkčnost jednotlivých modulů.

3.2.2.1 FUNKCE PRO ZPRACOVÁNÍ DOTAZŮ

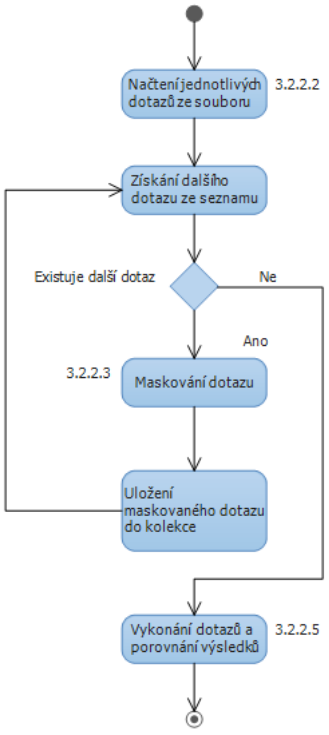
Vstupy: Připojení k původní databázi, připojení k databázi metadat, připojení k maskované databázi, vstupní soubor s dotazy

Výstupy: Množina maskovaných dotazů, ke každému dotazu náleží velikost výsledku původní i maskované verze a příznak, zda jsou jejich exekuční plány ekvivalentní

Hlavní funkce tohoto modulu a gro celé této práce. Vstupem této funkce je soubor s vytížením, připojení k původní databázi, metadatům a maskované databázi. Ze vstupního souboru jsou přečteny jednotlivé dotazy a uloženy do kolekce původních dotazů, viz 3.2.2.2 Funkce načtení dotazů ze souboru. Dotazy z kolekce jsou poté sekvenčně zpracovávány. Každý dotaz je poté maskován, viz 3.2.2.3 Funkce maskování dotazu. Zpracovaný dotaz je poté uložen do kolekce zpracovaných dotazů.

Až jsou všechny dotazy zpracovány, je pro každou dvojici (původní - maskovaný) z dotazů vykonáno porovnání velikosti výsledku a exekučního plánu, viz 3.2.2.5 Funkce porovnání velikosti výsledků a exekučních plánů.

Tato funkce již byla implementována, ale v rámci mé práce byla upravena tak, aby bylo možno její výstup použít v GUI.



Obrázek 8: Aktivity diagram maskování dotazů

3.2.2.2 FUNKCE NAČTENÍ DOTAZŮ ZE SOUBORU

Vstupy: Cesta k souboru s dotazy

Výstupy: Jednotlivé parsované dotazy pro další zpracování

Funkce načte a rozparsuje dotazy ze vstupního souboru a uloží je do úložistě dotazů, ze kterého jsou poté postupně zpracovávány a maskovány. Vstupní soubor může být ve formátu SQL, v tomto případě je požadavek na formát dotazů v souboru, aby byl každý dotaz deklarován jako VIEW. Tohoto formátu můžeme dosáhnout exportem již existujících VIEW z Microsoft SQL Serveru do souboru, nebo manuální úpravou. Druhý podporovaný formát vstupního souboru je XML, jedná se o vlastní formát XML souboru a podporuje i více kombinací hodnot parametrů pro jeden dotaz. Jednoduchá ukázka formátu XML souboru je níže.

```
<?xml version="1.0" encoding="UTF-8"?>
<workload>
  <sql id="1" text="select * from person where p_name = ?" paramcount="1" count="2"
type="dql">
    <values>
      <p>'John Smith'</p>
    </values>
    <values>
      <p>'Jane Doe'</p>
    </values>
  </sql>
</workload>
```

V textu dotazu je místo hodnoty každého parametru znak ‚?‘. Text dotazu je sekvenčně procházen a pokud narazíme na znak ‚?‘, nahradíme ho hodnotou z uzlu „values“ na odpovídajícím indexu. Tento formát má výhodu, pokud potřebujeme jeden dotaz testovat s různými hodnotami parametrů, nemusí se jeden dotaz v souboru několikrát opakovat. Tato funkce již byla implementována a mnou nebyla nijak modifikována, funguje od začátku spolehlivě.

3.2.2.3 FUNKCE MASKOVÁNÍ DOTAZU

Vstupy: Úložiště parsovaných dotazů, připojení k nemaskované databázi, připojení k metadatům

Výstupy: Množina maskovaných dotazů, ke každému dotazu náleží velikost výsledku původní i maskované verze a příznak, zda jsou jejich exekuční plány ekvivalentní

Jako první se načtou aktuální metadata pomocí 3.2.1.4 Funkce pro načtení metadat. Následně jsou sekvenčně analyzovány a maskovány jednotlivé dotazy. Všem existujícím a zatím nemaskovaným parametrům je přiřazena nová, unikátní hodnota. To proto, abychom mohli jednotlivé hodnoty jednoduše najít a identifikovat v plánu vykonání dotazu. Původní hodnoty parametrů jsou zálohovány a následně přiřazeny zpět. Dotaz s unikátními parametry je poté vykonán vůči původní databázi a relční databází je vrácen plán vykonání dotazu ve formátu XML, příkazem **SET SHOWPLAN_XML ON**.

Plán vykonání dotazu je analyzován na výskyt hodnot k maskování, tzn., jestli hodnota patří k nějakému atributu a pokud ano, tak k jakému, viz 3.2.2.4 Funkce analýzy dotazu. Nalezené hodnoty jsou ukládány do separátního úložiště, tzv., hodnoty zájmu. Každá taková hodnota je následně ověřena vůči

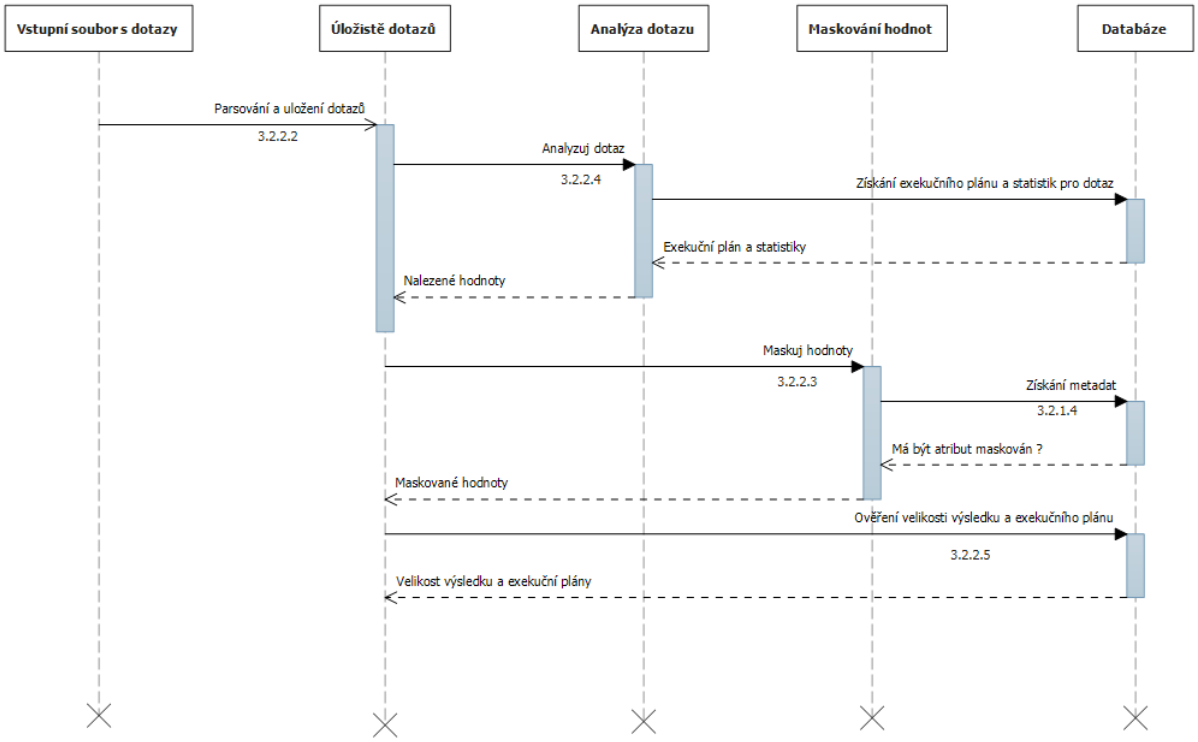
metadatům, zda byla v metadatech nalezena a pokud ano, jestli její datový typ odpovídá tomu v metadatech. Pokud je vše v pořádku, je postoupeno k maskování hodnot v dotazu. Pokud ovšem aplikace narazí na nesrovnalost, je provedena alternativní analýza dotazu.

Dotazu jsou první navráceny původní hodnoty parametrů, a to v obou případech, tedy ať už byla předchozí analýza úspěšná, či nikoliv. Alternativní analýza tedy spočívá v analýze dotazu bez unikátních parametrů a týká se pouze parametrizovaných dotazů, tzn. dotazů z XML souboru, zmíněného výše.

Po analýze dotazů již přichází na řadu maskování jednotlivých parametrů. Každý parametr je ověřen vůči metadatům, zda má dojít k jeho maskování. Logika maskování je samozřejmě stejná, jako v případě maskování samotné relační databáze, viz 3.2.1.3 Funkce maskování hodnoty. Až jsou maskovány všechny parametry, je takto maskovaný dotaz uložen do separátního úložiště zpracovaných dotazů.

Po maskování dotazu následuje poslední krok, a to získání velikosti výsledku pro oba dotazy, původní i maskovaný a také jejich exekuční plány. Výsledek tohoto kroku následně vidíme v uživatelském rozhraní a primárně slouží k určení úspěšnosti maskování.

Tato funkce již byla implementována, ale v rámci mé práce byly opravovány nalezené chyby nebo rozšiřována funkčnost. Byla opravena chyba rozpoznávání datového typu ‚datetime', byla přidána kolekce původní-maskovaný atributů a byla opravena chyba, kdy pro textové datové typy nebyla maskovaná hodnota ohraničena v jednoduchých uvozovkách.



Obrázek 9: Sekvenční diagram maskování dotazů

3.2.2.4 FUNKCE ANALÝZY DOTAZU

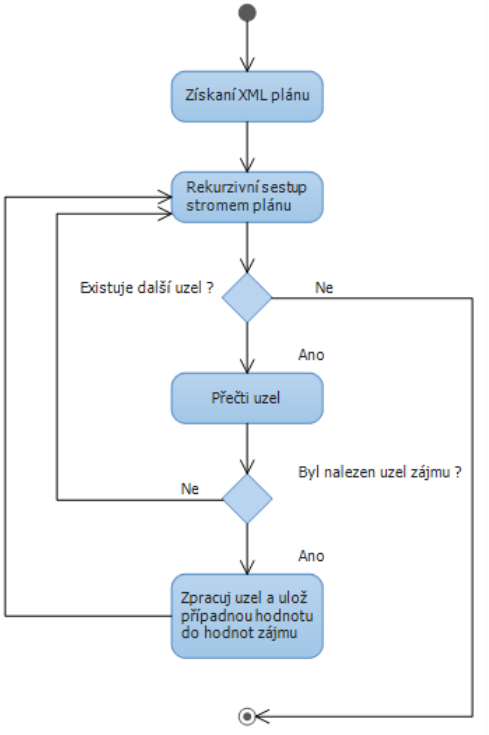
Vstupy: Dotaz z úložiště dotazů

Výstupy: Analyzovaný dotaz, tzn., v dotazu jsou nalezeny atributy a každý atribut má přiřazená metadata, pokud pro daný atribut existují

Analýza dotazu rekurzivním sestupem prochází exekuční plán ve formátu XML. Struktura a formát exekučního plánu jsou všeobecně známy. Známé jsou také operace (názvy jednotlivých uzlů v XML), můžeme tedy efektivně projít celý strom XML a vybrat z něj, pro nás důležité, informace.

Jako první se hledá uzel s názvem **QueryPlan**. Pokud najdeme zmíněný uzel, jsou v něm hledány uzly, které jsou pro nás důležité, tj. operace **ScalarOperator**, která indikuje operaci porovnání hodnot. To může být porovnání dvou konstant, porovnání dvou atributů, nebo pro nás nejduležitější, porovnání atributu s konstantou.

Pokud najdeme porovnání atributu s konstantou, je tato konstanta považována za hodnotu zájmu a uložena do kolekce hodnot zájmu společně s metadaty hodnoty, tj. název tabulky a atributu, ke kterému se tato hodnota váže. Hodnoty z kolekce zájmu jsou následně využívány k vyhodnocení, zda danou hodnotu maskovat či nikoliv. Tato funkce již byla implementována a v rámci mé práce jsem ji nerozšiřoval ani neupravoval. Případné chyby jsem, dle domluvy, oznamoval autorovi funkce, panu Radimu Bačovi.



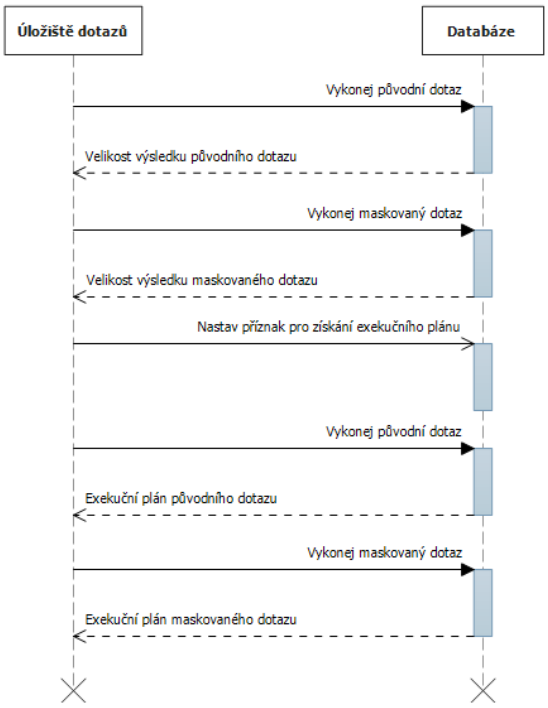
Obrázek 10: Aktivity diagram analýzy exekučního plánu

3.2.2.5 FUNKCE POROVNÁNÍ VELIKOSTI VÝSLEDKŮ A EXEKUČNÍCH PLÁNŮ

Vstupy: Připojení k původní databázi, připojení k maskované databázi, původní dotaz, maskovaný dotaz

Výstupy: Velikost výsledku původního i maskovaného dotazu, příznak, zda jsou jejich exekuční plány ekvivalentní

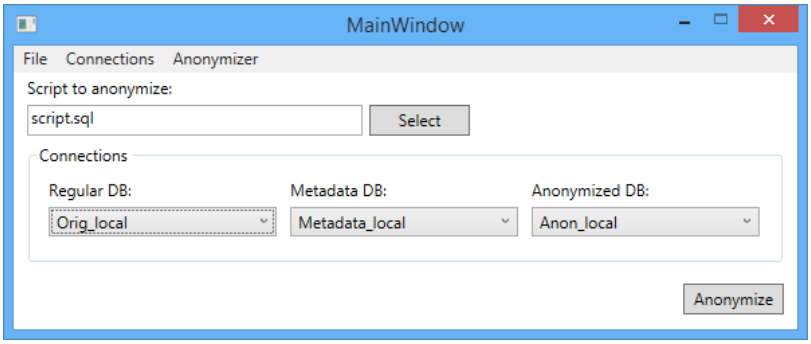
Poté, co jsou provedeny všechny úkony kolem maskování dotazu a dotaz je tedy maskován, je potřeba určit úspěšnost tohoto maskování. Úspěšnost můžeme vyjádřit např. porovnáním velikostí výsledku původního dotazu vůči maskovanému dotazu. Funkce tedy na vstupu očekává připojení k původní databázi, připojení k maskované databázi, původní dotaz a maskovaný dotaz. Dotazy jsou vykonány vůči databázím a je spočten počet vrácených řádků výsledku. Poté je nastaven příznak **SET SHOWPLAN_ALL ON** a dotazy jsou vykonány znovu. Tentokrát ovšem nepočítáme počet vrácených řádků, databáze vrátí zjednodušený exekuční plán pro každý dotaz. Tyto exekuční plány poté můžeme porovnat, zda jsou ekvivalentní. Tato funkce nebyla v původním modulu vůbec implementována.



Obrázek 11: Sekvenční diagram porovnání výsledků

3.2.3 UŽIVATELSKÉ ROZHRANÍ

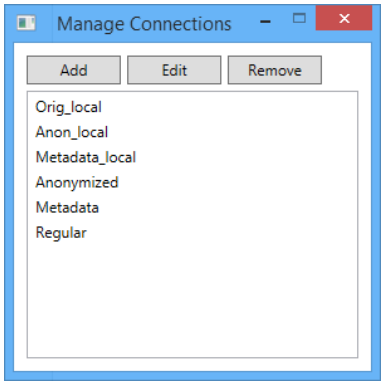
Uživatelské rozhraní spojuje Modul maskování databáze a Modul analýzy a maskování dotazu do jednoho celku. Poskytuje uživateli jednoduché rozhraní pro úkony spojené s maskováním relační databáze a jejího vytížení. Celé uživatelské rozhraní jsem implementoval v rámci mé práce sám.



Obrázek 12: Hlavní okno programu

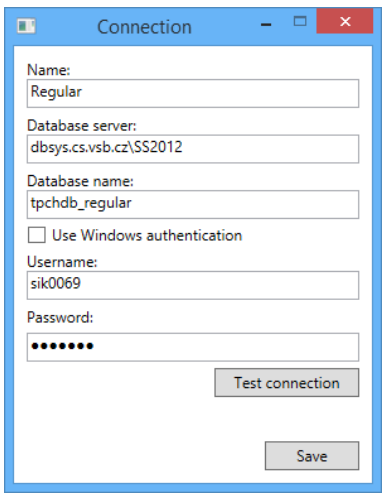
Uživatelské rozhraní sestává z menu a hlavního okna programu. Menu obsahuje položky **Connections**, pro správu uložených připojení k Microsoft SQL Serveru, dále **Anonymizer**, pro úkony spadající pod maskování databáze, tj. práce s metadaty a vlastní statické maskování.

Pokud z menu zvolíme položku **Connections->Manage connections**, zobrazí se okno se seznamem uložených připojení.



Obrázek 13: Uložená připojení

Připojení můžeme přidávat nebo upravovat a mazat již uložená. Po kliknutí na **Add** nebo **Edit** se zobrazí dialogové okno.

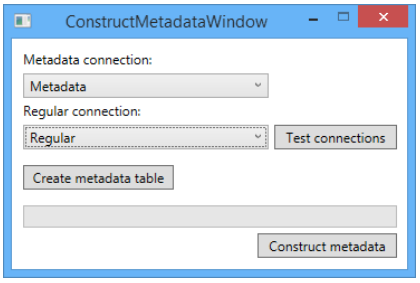


Obrázek 14: Dialogové okno připojení

Připojení můžeme pojmenovat, dále specifikujeme databázový server, název databáze a zvolíme způsob přihlašování. To může být buď autentikace pomocí Windows účtu, nebo pomocí uživatelského jména a hesla. Nakonec můžeme připojení otestovat a uložit.

3.2.3.1 FUNKCE VYTVOŘENÍ METADAT

Menu **Anonymizer->Construct metadata** zobrazí dialogové okno pro vytvoření metadat.

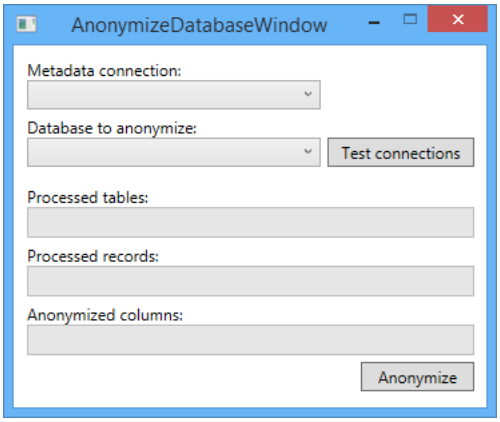


Obrázek 15: Dialogové okno metadat

Zde je nutno specifikovat připojení k databázi metadat a k originální, nemaskované, databázi. Zvolené připojení můžeme následně otestovat. Kliknutím na **Create metadata table** se, v databázi specifikované pomocí zvoleného připojení pod **Metadata connection**, smažou případné aktuální tabulky a metadata a vytvoří se nové, prázdné, struktury pro metadata. Stiskem tlačítka **Construct metadata** se následně vytvoří aktuální metadata pro databázi, specifikované pomocí zvoleného připojení pod **Regular connection**, využívá 3.2.1.1 Funkce pro vytvoření metadat.

3.2.3.2 FUNKCE MASKOVÁNÍ DATABÁZE

V menu **Anonymizer->Anonymize DB** již pouze zvolíme připojení k metadatům a cílovou databázi pro zamaskování.

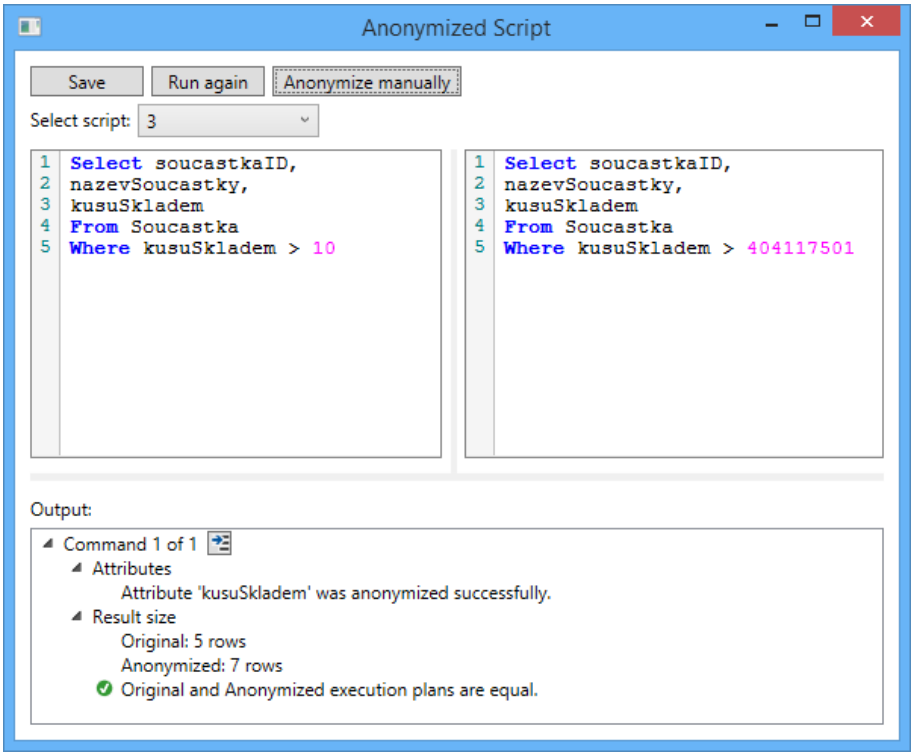


Obrázek 16: Dialogové okno maskování databáze

Opět můžeme zvolená připojení otestovat a pak už, kliknutím na tlačítko **Anonymize**, začne samotné maskování cílové databáze. Časová složitost se odvíjí od množství dat v cílové databázi k maskování, využívá 3.2.1.2 Funkce pro maskování databáze.

3.2.3.3 FUNKCE MASKOVÁNÍ VYTÍŽENÍ

Zpět k hlavní obrazovce (Obrázek 12: Hlavní okno programu), zde vybereme soubor obsahující dotazy k maskování, ve formátu SQL nebo XML. Dále vybereme připojení k jednotlivým databázím – původní nemaskované, metadata a maskované databázi. Poté stiskneme tlačítko **Anonymize**, které vykoná maskování dotazu. Po dokončení maskování se zobrazí následující okno s výsledkem maskování. Využívá 3.2.2.1 Funkce pro zpracování dotazů.

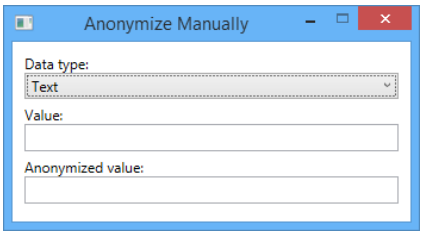


Obrázek 17: Okno maskovaného dotazu

Tlačítko **Save** – uloží zvolený sql příkaz do souboru

Tlačítko **Run again** – spustí zvolený skript znovu pro získání výsledků, např. po ruční úpravě dotazu

Tlačítko **Anonymize manually** – vyvolá dialogové okno pro manuální maskování parametrů, vhodné např. pro testování různých hodnot v dotazu nebo maskování hodnot, které nebyly maskovány automaticky. Využívá 3.2.3.4 Funkce ručního maskování hodnoty.



Obrázek 18: Dialogové okno pro manuální maskování

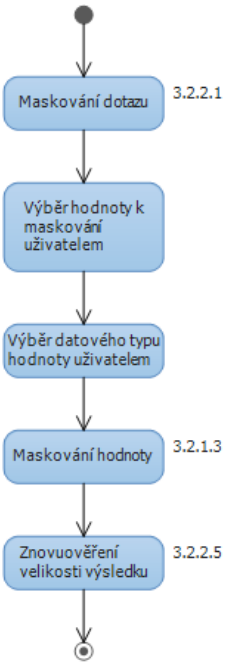
Výběr **Select script** – vybírá, se kterým z dotazů se aktuálně pracuje

Níže je vlevo původní dotaz a vpravo maskovaný dotaz. Pod dotazy je komponenta zobrazující výsledek maskování. V této komponentě jsou vypsány všechny nalezené hodnoty v dotazu, velikost výsledku pro původní i maskovaný dotaz, a zda jsou exekuční plány obou dotazů ekvivalentní.

3.2.3.4 FUNKCE RUČNÍHO MASKOVÁNÍ HODNOTY

Funkci můžeme zavolat dvěma způsoby. Buď kliknutím na tlačítko **Anonymize manually** v okně *Anonymized script*, viz Obrázek 17: Okno maskovaného dotazu. Druhá možnost je označit v komponentě maskovaného dotazu vybraný text, např. nemaskovaný atribut a poté přes kontextovou nabídku zvolit možnost **Anonymize selection**. Pokud zvolíme druhou možnost, v okně maskovaného dotazu se nám rovnou naplní textové pole Value vybranou hodnotou. Poté už pouze stačí vybrat správný datový typ. Pro maskování hodnoty se samozřejmě používá 3.2.1.3 Funkce maskování hodnoty, aby maskovaná hodnota odpovídala té v databázi.

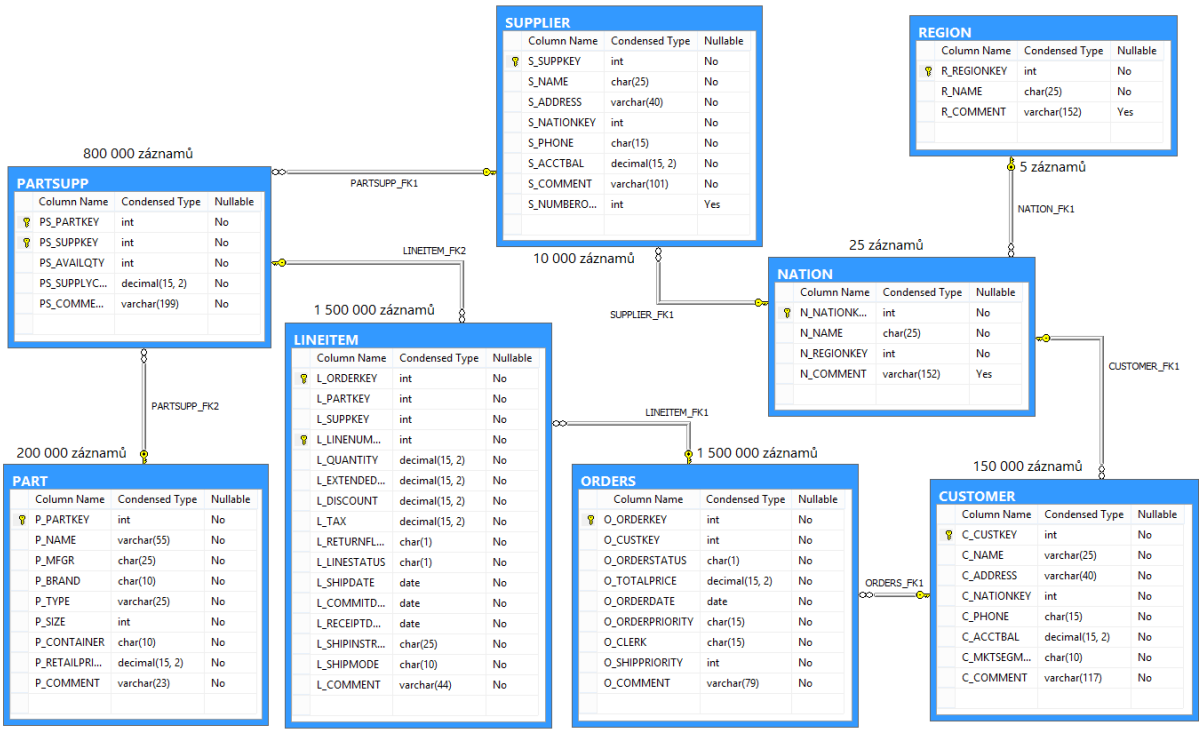
Po ručním maskování je vhodné znovu spustit dotaz tlačítkem **Run again** pro ověření velikosti výsledku.



Obrázek 19: Aktivity diagram ručního maskování

4 TESTOVÁNÍ

Testování aplikace probíhalo nad Microsoft SQL Serverem 2012. Během testování bylo zjištěno, že aplikace nepracuje správně pod Micorsoft SQL Serverem 2016, z důvodu změny formátu exekučního plánu dotazu. Jako testovací data byla použita databáze TPC-H s mírnými modifikacemi. Tyto modifikace spočívaly v odstranění redundantních hodnot, kvůli vytvoření primárních klíčů, které jsou pro chod aplikace nezbytné. Jako testovací dotazy posloužily dotazy z dokumentace TPC-H[5].



Obrázek 20: Schéma databáze TPC-H

4.1 TESTOVACÍ DOTAZY

V této kapitole probereme testovací dotazy, výsledky jejich maskování a případné problémy, které při maskování nastaly a proč. Dále také velikosti výsledků a porovnání exekučních plánů. V dokumentaci TPC-H je popsáno celkem 22 testovacích dotazů[6].

Musíme si také vysvětlit použité pojmy:

Původní velikost výsledku – velikost výsledku pro nemaskovaný dotaz vůči nemaskované databázi.

Velikost výsledku po maskování – velikost výsledku po automatickém maskování dotazu aplikací, bez zásahů uživatele, vůči maskované databázi.

Velikost výsledku po ručním domaskování – velikost výsledku po případném ručním domaskování hodnot, které aplikace automaticky nemaskovala (z nějakého důvodu aplikace hodnotu ignorovala, i když by neměla). Nebere se ohled na logičnost hodnot, tímto krokem pouze „simulujeme“ výstup aplikace, kdyby byla schopna najít všechny hodnoty k maskování spolehlivě.

Velikost výsledku po logickém ručním maskování - velikost výsledku po případné ruční úpravě hodnot, nejčastěji se jedná o prohození minima a maxima v rozsahové podmínce. Jde o teoreticky ideální stav, kdy aplikace detekuje a zamaskuje všechny hodnoty tak, jak by měla.

4.1.1 DOTAZ 1

```
select l_returnflag, l_linestatus, sum(l_quantity) as sum_qty, sum(l_extendedprice) as
sum_base_price, sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge, avg(l_quantity)
as avg_qty, avg(l_extendedprice) as avg_price, avg(l_discount) as avg_disc, count(*)
as count_order
from lineitem
where l_shipdate <= '1997-09-01'
group by l_returnflag, l_linestatus
order by l_returnflag, l_linestatus
```

Automatické maskování dotazu – neproběhne správně, atribut l_shipdate není maskován, aplikace, za určitých okolností nemaskuje datumy. Příčina zatím nebyla zjištěna. Pokud ručně zamaskujeme hodnotu `'1997-09-01'` dostaneme hodnotu `'1901-06-30'`.

Původní velikost výsledku – 4 záznamy

Velikost výsledku po maskování – 3 záznamy

Velikost výsledku po ručním domaskování – 1 záznam

Exekuční plán je ekvivalentní pro původní i maskovaný dotaz.

4.1.2 DOTAZ 2

```
select s_acctbal, s_name, n_name, p_partkey, p_mfgr, s_address, s_phone, s_comment
from part, supplier, partsupp, nation, region
where p_partkey = ps_partkey and s_suppkey = ps_suppkey and p_size = 15 and p_type
like '%BRASS' and s_nationkey = n_nationkey and n_regionkey = r_regionkey and r_name =
'EUROPE' and ps_supplycost =
       (select min(ps_supplycost)
        from partsupp, supplier, nation, region
        where p_partkey = ps_partkey and s_suppkey = ps_suppkey and s_nationkey =
n_nationkey and n_regionkey = r_regionkey and r_name = 'EUROPE')
order by s_acctbal desc, n_name, s_name, p_partke
```

Automatické maskování dotazu – neproběhne správně, atribut p_type není maskován, aplikace momentálně neumí maskovat hodnoty ve spojení s operátorem LIKE. Pokud ručně zamaskujeme hodnotu `'BRASS'` dostaneme hodnotu `'RbQcc'`. Znak ‚%‘ nemaskujeme.

Původní velikost výsledku – 461 záznamů

Velikost výsledku po maskování – 0 záznamů

Velikost výsledku po ručním domaskování – 486 záznamů

Exekuční plán je ekvivalentní pro původní i maskovaný dotaz.

4.1.3 DOTAZ 3

```
select l_orderkey, sum(l_extendedprice * (1 - l_discount)) as revenue, o_orderdate,
o_shippriority
from customer, orders, lineitem
where c_mktsegment = 'BUILDING' and c_custkey = o_custkey and l_orderkey = o_orderkey
and o_orderdate < '1995-03-15' and l_shipdate > '1995-03-15'
group by l_orderkey, o_orderdate, o_shippriority
order by revenue desc, o_orderdate
```

Automatické maskování dotazu – proběhne v pořádku.

Původní velikost výsledku – 43 703 záznamů

Velikost výsledku po maskování – 15 405 záznamů

Exekuční plán není ekvivalentní pro původní a maskovaný dotaz.

4.1.4 DOTAZ 4

```
select o_orderpriority, count(*) as order_count
from orders
where o_orderdate >= '1993-07-01' and o_orderdate < '1993-10-01' and exists
      (select * from lineitem where l_orderkey = o_orderkey and l_commitdate <
l_receiptdate)
group by o_orderpriority
order by o_orderpriority
```

Automatické maskování dotazu – neproběhne v pořádku, atribut o_orderdate není ani v jednom případě maskován.

Původní velikost výsledku – 5 záznamů

Velikost výsledku po maskování – 0 záznamů

Velikost výsledku po ručním domaskování – 5 záznamů

Exekuční plán je ekvivalentní pro původní i maskovaný dotaz.

4.1.5 DOTAZ 5

```
select n_name, sum(l_extendedprice * (1 - l_discount)) as revenue
from customer, orders, lineitem, supplier, nation, region
where c_custkey = o_custkey and l_orderkey = o_orderkey and l_suppkey = s_suppkey and
c_nationkey = s_nationkey and s_nationkey = n_nationkey and n_regionkey = r_regionkey
and r_name = 'ASIA' and o_orderdate >= '1994-01-01' and o_orderdate < '1995-01-01'
group by n_name
order by revenue desc
```

Automatické maskování dotazu – neproběhne v pořádku, atribut o_orderdate není ani v jednom případě maskován.

Původní velikost výsledku – 5 záznamů

Velikost výsledku po maskování – 0 záznamů

Velikost výsledku po ručním domaskování – 5 záznamů

Exekuční plán je ekvivalentní pro původní i maskovaný dotaz.

4.1.6 DOTAZ 6

```
select sum(l_extendedprice * l_discount) as revenue
from lineitem
where l_shipdate >= '1994-01-01' and l_shipdate < '1995-01-01' and l_discount between
0.06 - 0.01 and 0.06 + 0.01 and l_quantity < 24
```

Automatické maskování dotazu – neproběhne v pořádku, atribut l_shipdate není ani v jednom případě maskován, atribut l_discount není maskován a ani atribut l_quantity není maskován.

Původní velikost výsledku – 1 záznam

Velikost výsledku po maskování – 1 záznam

Velikost výsledku po ručním domaskování – 1 záznam

Exekuční plán není ekvivalentní pro původní a maskovaný dotaz.

4.1.7 DOTAZ 7

```
select supp_nation, cust_nation, l_year, sum(volume) as revenue
from
       (select n1.n_name as supp_nation, n2.n_name as cust_nation, year(l_shipdate) as
l_year, l_extendedprice * (1 - l_discount) as volume
        from supplier, lineitem, orders, customer, nation n1, nation n2
        where s_suppkey = l_suppkey and o_orderkey = l_orderkey and c_custkey =
o_custkey and s_nationkey = n1.n_nationkey and c_nationkey = n2.n_nationkey and
((n1.n_name = 'FRANCE' and n2.n_name = 'GERMANY') or (n1.n_name = 'GERMANY' and
n2.n_name = 'FRANCE')) and l_shipdate between '1995-01-01' and '1996-12-31') as
shipping
group by supp_nation, cust_nation, l_year
order by supp_nation, cust_nation, l_year
```

Automatické maskování dotazu – neproběhne v pořádku, atribut l_shipdate není maskován, ale pouze spodní hranice rozsahu. Po ručním maskování spodní hranice rozsahu je ale tato hodnota vyšší, než horní hranice rozsahu. Musíme proto hodnoty prohodit.

Původní velikost výsledku – 4 záznamy

Velikost výsledku po maskování – 0 záznamů

Velikost výsledku po ručním domaskování – 0 záznamů

Velikost výsledku po logickém ručním maskování – 8 záznamů

Exekuční plán není ekvivalentní pro původní a maskovaný dotaz.

4.1.8 DOTAZ 8

```
select o_year, sum(case when nation = 'BRAZIL' then volume else 0 end) / sum(volume)
as mkt_share
from (select year(o_orderdate) as o_year, l_extendedprice * (1 - l_discount) as
volume, n2.n_name as nation
        from part, supplier, lineitem, orders, customer, nation n1, nation n2, region
        where p_partkey = l_partkey and s_suppkey = l_suppkey and l_orderkey =
o_orderkey and o_custkey = c_custkey and c_nationkey = n1.n_nationkey and
n1.n_regionkey = r_regionkey and r_name = 'AMERICA' and s_nationkey = n2.n_nationkey
and o_orderdate between '1995-01-01' and '1996-12-31' and p_type = 'ECONOMY ANODIZED
STEEL')
       as all_nations
group by o_year
order by o_year
```

Automatické maskování dotazu – neproběhne v pořádku, atribut l_shipdate není maskován, ale pouze spodní hranice rozsahu. Po ručním maskování spodní hranice rozsahu je ale tato hodnota vyšší, než horní hranice rozsahu. Musíme proto hodnoty prohodit.

Původní velikost výsledku – 2 záznamy

Velikost výsledku po maskování – 0 záznamů

Velikost výsledku po ručním domaskování – 0 záznamů

Velikost výsledku po logickém ručním maskování – 4 záznamy

Exekuční plán není ekvivalentní pro původní a maskovaný dotaz.

4.1.9 DOTAZ 9

```
select nation, o_year, sum(amount) as sum_profit
from (select n_name as nation, year(o_orderdate) as o_year, l_extendedprice * (1 -
l_discount) - ps_supplycost * l_quantity as amount
        from part, supplier, lineitem, partsupp, orders, nation
        where s_suppkey = l_suppkey and ps_suppkey = l_suppkey and ps_partkey =
l_partkey and p_partkey = l_partkey and o_orderkey = l_orderkey and s_nationkey =
n_nationkey and p_name like '%green%')
       as profit
group by nation, o_year
order by nation, o_year desc
```

Automatické maskování dotazu – neproběhne v pořádku, atribut p_name není maskován, jelikož se vztahuje k operátoru LIKE.

Původní velikost výsledku – 175 záznamů

Velikost výsledku po maskování – 0 záznamů

Velikost výsledku po ručním domaskování – 175 záznamů

Exekuční plán není ekvivalentní pro původní a maskovaný dotaz.

4.1.10 DOTAZ 10

```
select c_custkey, c_name, sum(l_extendedprice * (1 - l_discount)) as revenue,
c_acctbal, n_name, c_address, c_phone, c_comment
from customer, orders, lineitem, nation
where c_custkey = o_custkey and l_orderkey = o_orderkey and o_orderdate >= '1993-10-
01' and o_orderdate < '1994-01-01' and l_returnflag = 'R' and c_nationkey =
n_nationkey
group by c_custkey, c_name, c_acctbal, c_phone, n_name, c_address, c_comment
order by revenue desc
```

Automatické maskování dotazu – neproběhne v pořádku, atribut o_orderdate není maskován ani v jednom případě a atribut l_returnflag také není, ze zatím neznámých důvodů, maskován. Pokud atributy domaskujeme manuálně, v rozsahové podmínce je přehozena spodní a horní hranice, musíme je tedy ručně prohodit.

Původní velikost výsledku – 19 072 záznamů

Velikost výsledku po maskování – 0 záznamů

Velikost výsledku po ručním domaskování – 0 záznamů

Velikost výsledku po logickém ručním maskování – 76 744 záznamů

Exekuční plán není ekvivalentní pro původní a maskovaný dotaz.

4.1.11 DOTAZ 11

```
select ps_partkey, sum(ps_supplycost * ps_availqty) as value
from partsupp, supplier, nation
where ps_suppkey = s_suppkey and s_nationkey = n_nationkey and n_name = 'GERMANY'
group by ps_partkey
having sum(ps_supplycost * ps_availqty) >
       (select sum(ps_supplycost * ps_availqty) * 0.0001 from partsupp, supplier,
nation
          where ps_suppkey = s_suppkey and s_nationkey = n_nationkey and n_name =
'GERMANY')
order by value desc
```

Automatické maskování dotazu – Proběhne bez problému.

Původní velikost výsledku – 1 013 záznamů

Velikost výsledku po maskování – 981 záznamů

Exekuční plán je ekvivalentní pro původní i maskovaný dotaz.

4.1.12 Dotaz 12

```
select l_shipmode, sum(case when o_orderpriority = '1-URGENT' or o_orderpriority = '2-
HIGH' then 1 else 0 end) as high_line_count, sum(case when o_orderpriority <> '1-
URGENT' and o_orderpriority <> '2-HIGH' then 1 else 0 end) as low_line_count
from orders, lineitem
where o_orderkey = l_orderkey and l_shipmode in ('MAIL', 'SHIP') and l_commitdate <
l_receiptdate and l_shipdate < l_commitdate and l_receiptdate >= '1994-01-01' and
l_receiptdate < '1995-01-01'
group by l_shipmode
order by l_shipmode
```

Automatické maskování dotazu – neproběhne v pořádku, atribut l_receiptdate není maskován ani v jednom případě.

Původní velikost výsledku – 2 záznamy

Velikost výsledku po maskování – 0 záznamů

Velikost výsledku po ručním domaskování – 2 záznamy

Exekuční plán není ekvivalentní pro původní a maskovaný dotaz.

4.1.13 Dotaz 13

```
select c_count, count(*) as custdist
from (select c_custkey, count(o_orderkey)
        from customer left outer join orders on c_custkey = o_custkey and o_comment
not like '%special%requests%'
        group by c_custkey) as c_orders (c_custkey, c_count)
group by c_count
order by custdist desc, c_count desc
```

Automatické maskování dotazu – neproběhne v pořádku, atribut o_comment není maskován, jelikož se jedná o operátor LIKE.

Původní velikost výsledku – 40 záznamů

Velikost výsledku po maskování – 40 záznamů

Velikost výsledku po ručním domaskování – 40 záznamů

Exekuční plán je ekvivalentní pro původní i maskovaný dotaz.

4.1.14 Dotaz 14

```
select 100.00 * sum(case when p_type like 'PROMO%' then l_extendedprice * (1 -
l_discount) else 0 end) / sum(l_extendedprice * (1 - l_discount)) as promo_revenue
from lineitem, part
where l_partkey = p_partkey and l_shipdate >= '1995-09-01' and l_shipdate < '1995-10-
01'
```

Automatické maskování dotazu – neproběhne v pořádku, atribut p_type není maskován, jelikož se jedná o operátor LIKE a l_shipdate není maskován ani v jednom z výskytů.

Původní velikost výsledku – 1 záznam (suma)

Velikost výsledku po maskování – 1 záznam (suma, ovšem hodnota je NULL)

Velikost výsledku po ručním domaskování – 1 záznam (suma, má hodnotu)

Exekuční plán je ekvivalentní pro původní i maskovaný dotaz.

4.1.15 DOTAZ 15

```
create view revenue_s (supplier_no, total_revenue) as
       select l_suppkey, sum(l_extendedprice * (1 - l_discount))
       from lineitem
       where l_shipdate >= '1996-01-01' and l_shipdate < '1996-04-01'
       group by l_suppkey

select s_suppkey, s_name, s_address, s_phone, total_revenue
from supplier, revenue_s
where s_suppkey = supplier_no and total_revenue = (select max(total_revenue) from
revenue_s)
order by s_suppkey

drop view revenue_s
```

Tento dotaz sestává z vytvoření pohledu, jeho využití v dotazu a následnému smazání pohledu a jako takový neprojde maskovací aplikací. Pohled by se musel vytvořit a maskovat v relační databázi a testovací dotaz by ho pak jen mohl využívat.

4.1.16 DOTAZ 16

```
select p_brand, p_type, p_size, count(distinct ps_suppkey) as supplier_cnt
from partsupp, part
where p_partkey = ps_partkey and p_brand <> 'BRAND#45' and p_type not like 'MEDIUM
POLISHED%' and p_size in (49, 14, 24, 45, 19, 3, 36, 9) and ps_suppkey not in (select
s_suppkey from supplier where s_comment like '%Customer%Complaints%')
group by p_brand, p_type, p_size
order by supplier_cnt desc, p_brand, p_type, p_size
```

Automatické maskování dotazu – neproběhne v pořádku, atributy p_type a s_comment nejsou maskovány, jelikož se jedná o operátory LIKE.

Původní velikost výsledku – 18 268 záznamů

Velikost výsledku po maskování – 19 696

Velikost výsledku po ručním domaskování – 19 061 záznamů

Exekuční plán je ekvivalentní pro původní i maskovaný dotaz.

4.1.17 Dotaz 17

```
select sum(l_extendedprice) / 7.0 as avg_yearly
from lineitem, part
where p_partkey = l_partkey and p_brand = 'Brand#23' and p_container = 'MED BOX' and
l_quantity < (select 0.2 * avg(l_quantity) from lineitem where l_partkey = p_partkey)
```

Automatické maskování dotazu – proběhne bez problémů.

Původní velikost výsledku – 1 záznam (suma)

Velikost výsledku po maskování – 1 záznam (suma)

Exekuční plán je ekvivalentní pro původní i maskovaný dotaz.

4.1.18 Dotaz 18

```
select c_name, c_custkey, o_orderkey, o_orderdate, o_totalprice, sum(l_quantity)
from customer, orders, lineitem
where o_orderkey in (select l_orderkey from lineitem group by l_orderkey having
sum(l_quantity) > 49) and c_custkey = o_custkey and o_orderkey = l_orderkey
group by c_name, c_custkey, o_orderkey, o_orderdate, o_totalprice
order by o_totalprice desc, o_orderdate
```

Automatické maskování dotazu – neproběhne v pořádku, hodnota 49 není maskována

Původní velikost výsledku – 30 147 záznamů

Velikost výsledku po maskování – 1 500 000

Velikost výsledku po ručním domaskování – 990 950 záznamů

Exekuční plán není ekvivalentní pro původní a maskovaný dotaz.

4.1.19 Dotaz 19

```
select sum(l_extendedprice* (1 - l_discount)) as revenue
from lineitem, part
where (p_partkey = l_partkey and p_brand = 'Brand#12' and p_container in ('SM CASE',
'SM BOX', 'SM PACK', 'SM PKG') and l_quantity >= 1 and l_quantity <= 1 + 10 and p_size
between 1 and 5 and l_shipmode in ('AIR', 'AIR REG') and l_shipinstruct = 'DELIVER IN
PERSON')
or (p_partkey = l_partkey and p_brand = 'Brand#23' and p_container in ('MED BAG', 'MED
BOX', 'MED PKG', 'MED PACK') and l_quantity >= 10 and l_quantity <= 10 + 10 and p_size
between 1 and 10 and l_shipmode in ('AIR', 'AIR REG') and l_shipinstruct = 'DELIVER IN
PERSON')
or (p_partkey = l_partkey and p_brand = 'Brand#34' and p_container in ('LG CASE', 'LG
BOX', 'LG PACK', 'LG PKG') and l_quantity >= 20 and l_quantity <= 20 + 10 and p_size
between 1 and 15 and l_shipmode in ('AIR', 'AIR REG') and l_shipinstruct = 'DELIVER IN
PERSON')
```

Automatické maskování dotazu –atributy l_quantity nejsou maskovány, nicméně v tomto případě nemusí jít nutně o chybu. Jedná se o rozsahové podmínky, kdy je spodní hranice určena hodnotou a horní hranice je hodnota spodní hranice plus 10. Aplikace nemůže vědět, zda má maskovat obě hodnoty,

či nikoliv. Při testu s ručním maskováním obou hodnot, tzn., byla maskována hodnota 1 a hodnota 10 separátně a poté byly sečteny, došlo k přetečení hodnoty INT. Rozsah by byl příliš velký a velikost výsledku by byla tudíž také.

Aplikace ovšem nemůže poznat, že by měla maskovat pouze jednu hodnotu, musí zasáhnout uživatel, a maskovat dotaz ručně.

Původní velikost výsledku – 1 záznam (suma)

Velikost výsledku po maskování – 1 záznam (suma, hodnota je NULL)

Velikost výsledku po ručním domaskování – 1 záznam (suma, má hodnotu)

Exekuční plán je ekvivalentní pro původní i maskovaný dotaz.

4.1.20 Dotaz 20

```
select s_name, s_address
from supplier, nation
where s_suppkey in (select ps_suppkey from partsupp where ps_partkey in (select
p_partkey from part where p_name like 'forest%') and ps_availqty > (select 0.5 *
sum(l_quantity) from lineitem where l_partkey = ps_partkey and l_suppkey = ps_suppkey
and l_shipdate >= '1994-01-01' and l_shipdate < '1995-01-01'))
       and s_nationkey = n_nationkey and n_name = 'CANADA'
order by s_name
```

Automatické maskování dotazu – neproběhne v pořádku, atribut p_name není maskován, jelikož se jedná o LIKE a l_shipdate také není maskováno.

Původní velikost výsledku – 83 záznamů

Velikost výsledku po maskování – 0 záznamů

Velikost výsledku po ručním domaskování – 0 záznamů (problém s enkódováním znaků za hodnotou ASCII)

Velikost výsledku po logickém ručním maskování – 109 záznamů

Exekuční plán není ekvivalentní pro původní a maskovaný dotaz.

4.1.21 Dotaz 21

```
select s_name, count(*) as numwait
from supplier, lineitem l1, orders, nation
where s_suppkey = l1.l_suppkey and o_orderkey = l1.l_orderkey and o_orderstatus = 'F'
and l1.l_receiptdate > l1.l_commitdate and exists
(select * from lineitem l2 where l2.l_orderkey = l1.l_orderkey and l2.l_suppkey <>
l1.l_suppkey)
and not exists (select * from lineitem l3 where l3.l_orderkey = l1.l_orderkey and
l3.l_suppkey <> l1.l_suppkey and l3.l_receiptdate > l3.l_commitdate)
and s_nationkey = n_nationkey and n_name = 'SAUDI ARABIA'
group by s_name order by numwait desc, s_name
```

Automatické maskování dotazu – proběhne v pořádku, ovšem dotaz nevrací žádné výsledky.

Původní velikost výsledku – 0 záznamů

Velikost výsledku po maskování – 0 záznamů

Exekuční plán je ekvivalentní pro původní i maskovaný dotaz.

4.1.22 DOTAZ 22

```
select cntrycode, count(*) as numcust, sum(c_acctbal) as totacctbal
from (select substring(c_phone, 1 , 2) as cntrycode, c_acctbal
         from customer
         where substring(c_phone, 1 , 2) in ('13', '31', '23', '29', '30', '18', '17')
and c_acctbal >
              (select avg(c_acctbal) from customer where c_acctbal > 0.00 and
substring(c_phone, 1, 2) in ('13', '31', '23', '29', '30', '18', '17'))
           and not exists (select * from orders where o_custkey = c_custkey)
       ) as custsale
group by cntrycode
order by cntrycode
```

Automatické maskování dotazu – proběhne v pořádku.

Původní velikost výsledku – 7 záznamů

Velikost výsledku po maskování – 7 záznamů

Exekuční plán je ekvivalentní pro původní i maskovaný dotaz.

4.2 ZHODNOCENÍ TESTOVÁNÍ

Testování proběhlo, co se základní funkčnosti týče, až na jeden dotaz bez problémů. Aplikace tedy byla schopna tyto dotazy analyzovat a zpracovat. Úroveň úspěšnosti jejich maskování už se ale razantně liší.

Největší problémy aktuálně způsobují nemaskované hodnoty v dotazech, nejčastěji datumy, a operátor LIKE.

Musíme si také určit, kdy je ještě velikost výsledku maskovaného dotazu vůči velikosti výsledku původního dotazu považována za přijatelnou. U dotazů s původní velikostí výsledku do 100 záznamů můžeme brát jako úspěch, pokud je velikost výsledku maskovaného dotazu ± 50 záznamů. Pro větší velikost výsledku už můžeme jako úspěch uvažovat, pokud jsou velikostí výsledku řádově stejné.

V Tabulka 1: Souhrn výsledků maskování dotazů tedy můžeme vidět všechny výsledky. Je zřejmé, že aktuální stav je zatím stále nedostatečný, správně je maskováno pouze 10 z 22 dotazů. Z toho jeden dotaz aplikace vůbec nedokáže zpracovat. Po odlazení nejzávažnějších chyb, tj. chyby popsané v 5.1 Problémy, se už dostaneme na úspěšnost 16 z 22 maskovaných dotazů, což už se dá považovat za úspěch. Uživatel už se poté musí věnovat pouze pár dotazům, které bude muset maskovat ručně.

Pokud by se povedlo odladit i rozsahové podmínky tak, jak je popsáno v 5.2.1 Ošetření rozsahových podmínek, zůstane pro ruční domaskování pouze jediný dotaz, a to 4.1.19 Dotaz 19, kde nelze jednoznačně určit, jak maskovat tyto rozsahové podmínky.

Číslo dotazu	(1)	(2)	(3)	(4)	Komentář
1	4	3	1	Není potřeba	V pořádku
2	461	0	486	Není potřeba	V pořádku
3	43 703	0	15 000	Není potřeba	V pořádku
4	5	0	5	Není potřeba	V pořádku
5	5	0	5	Není potřeba	V pořádku
6	1	1	1	Není potřeba	V pořádku
7	4	0	0	8	V pořádku
8	2	0	0	4	V pořádku
9	175	0	175	Není potřeba	V pořádku
10	19 072	0	0	76 744	V pořádku
11	1 013	981	Není potřeba	Není potřeba	V pořádku
12	2	0	2	Není potřeba	V pořádku
13	40	40	40	Není potřeba	V pořádku
14	1	1	1	Není potřeba	V pořádku
15	Nefunguje	Nefunguje	Nefunguje	Nefunguje	Nefunguje
16	18 268	19 696	19 061	Není potřeba	V pořádku
17	1	1	Není potřeba	Není potřeba	V pořádku
18	30 147	1 500 000	990 950	Není potřeba	Příliš mnoho záznamů
19	1	1	1	Není potřeba	V pořádku
20	83	0	0	109	V pořádku
21	0	0	0	Není potřeba	V pořádku
22	7	7	Není potřeba	Není potřeba	V pořádku
Úspěšnost	21/22	10/22	16/22	20/22	

Tabulka 1: Souhrn výsledků maskování dotazů

(1) Původní velikost výsledku
(2) Velikost výsledku po automatickém maskování
(3) Velikost výsledku po ručním domaskování
(4) Velikost výsledku po logickém ručním maskování

5 PROBLÉMY, MOŽNÁ VYLEPŠENÍ A BUDOUCÍ VÝVOJ

Aplikace je stále považována za koncept a můžeme říct, že je momentálně ve fázi alfa. Stále je potřeba spousta vývoje, aby byla aplikace použitelná v produkčním prostředí. Z toho vyplývá potřeba zmínit zde nejen aktuální stav a vývoj aplikace, ale zhodnotit i to, co je potřeba udělat dále.

Zhodnocení nedostatků aktuální implementace je jeden z přínosů této práce. Podařilo se simulovat několik situací, které nejsou aktuální verzí aplikace jednoduše řešitelné, a je navrženo řešení, které umožní tyto situace v budoucnu řešit a řešit i konceptuálně podobné problémy.

5.1 PROBLÉMY

Během testování nad databází TPC-H bylo zjištěno mnoho, zatím neznámých a neobjevených, problémů. Tyto problémy je potřeba řešit jako první, jelikož nejvíce brání a obtěžují při používání aplikace.

5.1.1 NEMASKOVÁNÍ NĚKTERÝCH ATRIBUTŮ

Jak bylo zmíněno výše mezi testovacími dotazy, často se stává, že u některého atributu zůstane původní, nemaskovaná, hodnota. V případě konstant, nevztahujících se k žádnému atributu, to jinak nejde, zde to musíme nechat na uživateli, aby si zvolil, zda danou hodnotu maskovat nebo ne, z dotazu nemůžeme poznat kontext konstanty.

V případě hodnot vztahujícím se k atributům je již potřeba tuto situaci řešit. Zvlášť v případě rozsáhlých dotazů se může lehce stát, že uživatel nějakou nemaskovanou hodnotu lehce přehlédne. Nápovědou zde může být velký rozdíl ve velikosti výsledku původního a maskovaného dotazu, ale ani to nemusí být za všech okolností spolehlivé. Jednoduše může nastat situace, kdy dotaz s jednou (nebo více) nemaskovanými hodnotami vrací přibližně stejný počet výsledků, ale v případě změny hodnot atributů již bude rozdíl ve velikosti výsledku značný. Nejzrádnější jsou v tomto případě dotazy obsahující funkce **sum**, **count**, atd., jelikož často vracejí správný počet výsledků, ale spočtené hodnoty mohou být NULL, což z aplikace neodhalíme.

Problém nejčastěji nastává pro atributy datumů a rozsahové podmínky. Důvody jsou zatím neznámé, ale bude se neprodleně řešit.

5.1.2 MASKOVÁNÍ TEXTU

Problém aktuální maskovací funkce pro textové datové typy. Maskovací funkce je momentálně schopna generovat, pro jednotlivé znaky, hodnoty ‚0' až ‚255' (decimálně), tzn., může generovat i netisknutelné znaky (hodnota znaku decimálně je menší než 32) nebo naopak hodnoty za ASCII (hodnota znaku decimálně je větší než 127).

Obojí je problém, netisknutelné znaky, jak vyplývá z jejich názvu, nejsou vidět a představují problém, např. pro ruční maskování hodnoty. Může se třeba stát, že hodnota obsahuje mezi viditelnými znaky ještě netisknutelné a pokud takovou hodnotu pouze opíšeme tak, jak jí vidíme, je jasné, že to nebude odpovídat hodnotě uložené v databázi.

Naopak hodnoty za ASCII představují problém při rozdílném enkódovaní textu mezi systémem a cílovou databází, kdy znaky se stejnou decimální hodnotou reprezentují jiný znak v systému a jiný v databázi.

Vzhledem k tomu, že funkce aktuálně generuje libovolný znak, může také nastat situace, kdy bude hodnota pro operátor LIKE obsahovat více znaků ‚%', než původně obsahovala, např. někde uprostřed textu a tímto bude změněn princip vyhledávání.

Problémy se dají relativně jednoduše řešit, ale spíše se do budoucna počítá rovnou se změnou maskovací funkce za lepší a bezpečnější.

5.1.3 OPERÁTOR LIKE

S operátorem LIKE se vážou dva problémy, z toho jeden by měl být relativně lehce řešitelný.

Prvním problémem je, že operátor LIKE není vůbec detekován analýzou dotazu, tudíž hodnota vázající se k atributu pomocí operátoru like je považována za konstantu a není maskována. Jedná se tedy o chybu v aplikaci.

Druhý problém je spíše úvahou k zamyšlení. Jde o to, jaké situace můžou nastat s operátorem LIKE a o to, jak fungují bezpečné maskovací funkce.

Každá bezpečná maskovací funkce musí fungovat na základě předchozího kontextu, tzn., že výsledná hodnota závisí na předchozích znacích. Za bezpečnou funkci nelze považovat pouze jednoduchou záměnu znaků.

Situace 1 – vyhledávání podle začátku slova

Hledáme v textu podle jeho začátku a nezajímá nás, co se vyskytuje za ním. Hodnota pro vyhledávání je ukončena znakem procenta. Např. p_name LIKE ‚Joe%'. Zde není žádný problém, jelikož známe kontext od začátku textu a hodnota ‚Joe' bude správně maskována. Velikost výsledku můžeme očekávat přibližně stejnou jako v případě nemaskovaného dotazu.

Situace 2 – vyhledávání podle konce slova

Zde již nastává problém, jelikož nemůžeme znát předchozí kontext slova a hlavně se tento kontext bude pro různé prefixy měnit. Uvažujme podmínku p_name LIKE ‚%Joe' a demonstrujeme si na příkladech možné sitauce.

Originální text	Maskovaný text
‚Joe is bad‘	‚rFdkBnkoiV‘
‚I like Joe‘	‚kHndOrsled‘
‚I hate Joe‘	‚kHWdecXP_b‘

Hodnota ‚Joe‘ je, jako samostatné slovo nebo pokud tímto slovem text začíná, maskován jako ‚rFd‘. Jak se ale zachovat v případě, že text tímto slovem nezačíná. Jak můžeme vidět, v těchto případech je slovo ‚Joe‘ pkaždé maskováno jako jiná sekvence znaků. Toto není možné jednoznačně určit a je potřeba s tím počítat. V případě dotazů obsahujících operátor LIKE, kdy nehledáme podle začátku slova z principu nelze zaručit správnost výsledku.

5.2 MOŽNÁ VYLEPŠENÍ

Vylepšení nejsou pro chod aplikace nezbytně důležitá, spíše zpříjemňují a usnadňují její používání. Jejich implementace tedy nemá takovou prioritu, jako oprava chyb, ale bylo by vhodné je implementovat poté, co budou opraveny chyby.

5.2.1 OŠETŘENÍ ROZSAHOVÝCH PODMÍNEK

Momentálně nejsou rozsahové podmínky nijak detekovány. Tyto podmínky mohou být zapsány dvěma způsoby. První je pomocí operátoru **AND**, tzn., *atribut_n > minimum AND atribut_n < maximum*, nebo pomocí operátoru **BETWEEN**, tzn., *atribut_n BETWEEN minimum AND maximum*.

Nyní může a v testovacích dotazech jsme viděli, že opravdu nastává situace, že minimum je větší, než maximum. Pokud budeme schopni jednoznačně identifikovat rozsahovou podmínku, můžeme poté rovnou ověřit, že minimum není větší, než maximum. Pokud bychom tento stav identifikovali, mohli bychom na něj patřične reagovat. Buď automatickým prohozením minima a maxima, nebo přinejmenším upozorněním uživatele.

5.2.2 MOŽNOST VOLBY MASKOVANÝCH SLOUPCŮ

Může nastat situace, kdy nechceme maskovat celou databázi, např. kvůli časové složitosti nebo kvůli nemožnosti ideálně maskovat některý z dotazů, např. kvůli operátoru LIKE, viz 5.1.3 Operátor like. Bylo by tedy vhodné dát uživateli možnost volby, před samotným maskováním databáze, zda chce automaticky maskovat všechny sloupce, nebo si uživatel sám vybere, které sloupce maskovat a které ne.

Také se můžeme setkat s primárním nebo cizím klíčem, který by bylo potřeba maskovat, tzn., nejednalo by se o číselník, ale o nějaké konkrétní hodnoty. V tomto případě by zase bylo vhodné mít možnost nastavit, že chceme maskovat tento sloupec i přesto, že se jedná o klíč. Tabulka metadat attributes by se tedy rozrostla o jeden sloupec, který by indikoval, zda bylo na daný sloupec aplikováno maskování.

5.2.3 VÍCE MASKOVACÍCH FUNKCÍ S MOŽNOSTÍ VOLBY

Nehledě na aktuální stav maskovacích funkcí by bylo dobré mít na výběr z vícero typů maskování pro jednotlivé datové typy. To by mohlo zahrnovat například:

- Pro textové datové typy – maskování náhodnými znaky, maskování s určitým formátem (např. email, číslo kreditní karty, telefonní číslo, atd…), maskování reálnými slovy (možnost zachovat počet znaků nebo počet slov) atp.
- Pro numerické datové typy – náhodné maskování, maskování se zachováním řádu (tak, aby se např. z čísla 50 nestal milion, ale číslo mezi 10 až 100)[7]
- Pro datové typy datumu – náhodné maskování, maskování se zachováním řádu (např. maskovat pouze čas a den, zachovat měsíc a rok)[7]

5.2.4 KOMPATIBILITA S SQL SERVEREM 2016

Během mé práce zrovna došlo k odstavení a přeinstalaci testovacího SQL Serveru, konkrétně školního „dbedu“. Na tomto serveru do té doby běžel Microsoft SQL Server 2012 a testování probíhalo v pořádku. Po přeinstalaci ovšem byl nainstalován Microsoft SQL Server 2016 a v aplikaci začalo docházet k chybám analýzy dotazu. Po konzultaci s vedoucím bylo zjištěno, že s novou verzí se změnil formát XML exekučního plánu dotazu. Nepomohlo ani nastavení SQL Serveru 2016 do režimu kompatibility s verzí 2012. Abychom se tím nemuseli zdržovat od práce, na server byla nainstalována další instance Microsoft SQL Serveru, opět ve verzi 2012.

Aplikace momentálně pouze zobrazí varování, že nepodporuje novější verzi SQL Serveru než 2012. Je ovšem v plánu doimplementovat podporu novějších verzí SQL Serveru.

5.3 DALŠÍ VÝVOJ

Dalším vývojem rozumíme netriviální rozšíření funkčnosti aplikace. Vývoj může probíhat zároveň s body ze sekce 5.2 Možná vylepšení, jelikož funkce jsou na sobě nezávislé.

5.3.1 ULOŽENÉ PROCEDURY, FUNKCE A POHLEDY

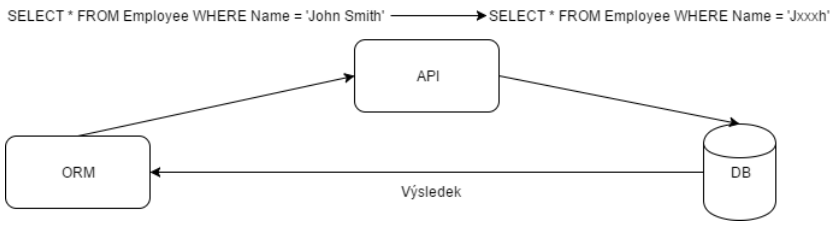
Kromě samotné databáze a testovacího vytížení existují v databázích entity, které je také potřeba maskovat. Jedná se především o pohledy, funkce a uložené procedury. V těchto entitách se mohou vyskytovat hodnoty atributů, které je pro správnou funkčnost potřeba také maskovat. Například se může jednat o procedury a funkce, které ve své logice využívají nějakého příznaku (třeba aktuální stav objednávky). Taková funkce nebo procedura by po maskování nefungovala správně.

Tento bod měl být i hlavní částí této práce, nicméně, vzhledem k množství objevených problémů s aplikací při testování nad TPC-H, byl odložen na později. Dokud samotné maskování dotazů nefunguje spolehlivě, není možné zabývat se maskováním funkčních entit databáze.

5.3.2 VÝVOJ API

Další možností, jak vývojáři zjednodušit práci, je poskytnutí komplexního API pro maskování dotazů. Tímto API by pak mohly být maskovány dotazy přímo na úrovni aplikace v datové vrstvě. Toto je ovšem možno řešit až po tom, co bude aplikace fungovat co nejvíce spolehlivě, protože řešit aktuální problémy maskování dotazu na úrovni aplikace je posléze v podstatě nemožné. Není možné koncovému uživateli při každém nesprávně maskovaném dotazu dávat na výběr, jak naložit s nemaskovanými hodnotami.

Navíc nelze jednoduše strojově určit, zda byl dotaz maskován úspěšně a pokud ne, které atributy by měly být ručně maskovány. Koncový uživatel také nejspíše vůbec nebude znalý konkrétní databáze a nakonec by si s problémem stejně nedokázal poradit.



Obrázek 21: API

6 ZÁVĚR

V první řadě je potřeba zmínit, že se nepovedlo naplnit zadání této práce jako takové. Důvodem bylo a stále je, že bylo objeveno množství chyb a nečekaných situací při testování složitějších dotazů. Samotné moduly, které jsem převzal, do té chvíle nikdy nebyly testovány nad TPC-H databází, kterou vedoucí určil jako laťku, pro určení přijatelné úspěšnosti maskování. Moduly samozřejmě byly zkoušeny i předtím, ale pouze pro jednodušší dotazy.

Aktuální úspěšnost automatického maskování je 10 z 22 dotazů, což je méně než polovina, a to je nedostatečné k tomu, aby se vývoj posunul dále, k maskování ještě složitějších entit, jakými T-SQL kód nepochybně je. I tak byl odveden slušný kus práce a aplikaci jsme, společně s vedoucím, posunuli blíže k praktickému použití. Jsme také domluveni, že vývoj bude pokračovat dále a pokusíme se, společnými silami, dovést aplikaci do stavu, kdy prakticky použitelná opravdu bude. Počáteční úspěšnost byla 4 z 22 dotazů. Takto velká neúspěšnost automatického maskování nebyla vedoucím předvídána, a tudíž nebylo v jeho možnostech podle toho upravit zadání.

Samotné výsledky maskování jako takového ovšem hodnotím velice pozitivně. Ať už výsledky dotazů, které byly maskovány korektně nebo dotazů, které byly domaskovány ručně jsou velice přesvědčivé. Velikosti výsledků maskovaných dotazů, až na jeden dotaz, se velmi blíží velikostem výsledků původních dotazů. Přístup je tedy zjevně správný, jen je potřeba doladit nalezené chyby.

Aplikace tedy zatím stále není použitelná pro praktické využití. Ovšem mohla by mít slušný potenciál, až bude dokončena. Existuje nespočet nástrojů pro maskování databáze, žádný ale neřeší, společně s maskováním samotné databáze, i maskování vytížení a případně ostatních entit v databázi, které je vhodné maskovat také. Takto komplexní maskovací nástroj totiž na trhu, zatím, neexistuje. Už jen z tohoto důvodu by stálo za to, aplikaci dokončit.

Na závěr bych rád poděkoval mému vedoucímu, panu Ing. Radimu Bačovi, Ph.D., za trpělivost, vstřícnost a pomoc při vedení této práce.

SEZNAM OBRÁZKŮ

Obrázek 1: SQL Server DDM (dynamické maskování) 6
Obrázek 2: CX-Mask (statické maskování) 7
Obrázek 3: Statické maskování (bez maskování dotazu) 8
Obrázek 4: Statické maskování (včetně maskování dotazu) 9
Obrázek 5: Schéma tabulky s metadaty 10
Obrázek 6: Aktivity diagram vytvoření metadat 11
Obrázek 7: Aktivity diagram maskování databáze 12
Obrázek 8: Aktivity diagram maskování dotazů 14
Obrázek 9: Sekvenční diagram maskování dotazů 16
Obrázek 10: Aktivity diagram analýzy exekučního plánu 17
Obrázek 11: Sekvenční diagram porovnání výsledků 18
Obrázek 12: Hlavní okno programu 19
Obrázek 13: Uložená připojení 19
Obrázek 14: Dialogové okno připojení 19
Obrázek 15: Dialogové okno metadat 20
Obrázek 16: Dialogové okno maskování databáze 20
Obrázek 17: Okno maskovaného dotazu 21
Obrázek 18: Dialogové okno pro manuální maskování 21
Obrázek 19: Aktivity diagram ručního maskování 22
Obrázek 20: Schéma databáze TPC-H 23
Obrázek 21: API 39

SEZNAM POUŽITÉ LITERATURY A ZDROJŮ

[1] REGER, Ronit. *Use Dynamic Data Masking to obfuscate your sensitive data* [online]. [cit. 2017-03-30]. Dostupné z: https://blogs.technet.microsoft.com/dataplatforminsider/2016/01/25/use-dynamic-data-masking-to-obfuscate-your-sensitive-data/

[2] *Why Static Data Masking is Not Enough* [online]. [cit. 2017-04-01]. Dostupné z: http://www.hexatier.com/dynamic-vs-static-data-masking-2/

[3] *Dynamic Data Masking* [online]. [cit. 2017-04-01]. Dostupné z: https://docs.microsoft.com/en-us/sql/relational-databases/security/dynamic-data-masking

[4] *Dynamic Data Masking* [online]. [cit. 2017-04-02]. Dostupné z: https://datamasking.com/wp-content/uploads/2016/06/CAMOUFLAGE-PRODUCTSHEET-CX-MASK.pdf

[5] *TPC BENCHMARK H Standard Specification Revision 2.17.1* [online]. [cit. 2017-04-15]. Dostupné z: http://www.tpc.org/tpc_documents_current_versions/pdf/tpc-h_v2.17.1.pdf

[6] *TPC BENCHMARK H Standard Specification Revision 2.17.1* [online]. , 29-67 [cit. 2017-04-15]. Dostupné z: http://www.tpc.org/tpc_documents_current_versions/pdf/tpc-h_v2.17.1.pdf

[7] *Data Masking: What You Need to Know: Number and Date Variance* [online]. , 9 [cit. 2017-04-17]. Dostupné z: http://www.datamasker.com/DataMasking_WhatYouNeedToKnow.pdf

PŘÍLOHY

Příloha na CD/DVD

- Zdrojové kódy aplikace
- Testovací dotazy
- Informace pro připojení k testovací databázi